

UNIVERSAL ASSEMBLER VERSION 1.2 JANUARY 4, 1978 (IN-HOUSE)

CONFIDENTIAL PROPRIETARY INFORMATION

THIS ITEM IS THE PROPERTY OF DATAPoint CORPORATION, SAN ANTONIO, TEXAS, AND CONTAINS CONFIDENTIAL AND TRADE SECRET INFORMATION. THIS ITEM MAY NOT BE TRANSFERRED FROM THE CUSTODY OR CONTROL OF DATAPoint EXCEPT AS AUTHORIZED BY DATAPoint AND THEN ONLY BY WAY OF LOAN FOR LIMITED PURPOSES. IT MUST NOT BE REPRODUCED IN WHOLE OR IN PART AND MUST BE RETURNED TO DATAPoint UPON REQUEST AND IN ALL EVENTS UPON COMPLETION OF THE PURPOSE OF THE LOAN.

NEITHER THIS ITEM NOR THE INFORMATION IT CONTAINS MAY BE USED OR DISCLOSED TO PERSONS NOT HAVING A NEED FOR SUCH USE OR DISCLOSURE CONSISTENT WITH THE PURPOSE OF THE LOAN, WITHOUT THE PRIOR WRITTEN CONSENT OF DATAPoint.

PAGE 2 1800MACR/TXT

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44

COMMAND LINE WAS: SNAP3 1800MACR,,,1800MACR/APLX

INCLUSION A: 1800/MAC
 INCLUSION B: MDEF1800/TXT
 INCLUSION C: BDEF1800/TXT
 INCLUSION D: ODEF1800/TXT
 INCLUSION E: STRUCT/TXT
 INCLUSION F: 1800REST/TXT
 INCLUSION G: 1800BOOT/TXT
 INCLUSION H: 1800DIAG/TXT
 INCLUSION I: 1800DSKR/TXT
 INCLUSION J: 1800DRUG/TXT
 INCLUSION K: 1800DSPR/TXT
 INCLUSION L: 1800MOVI/TXT

PROGRAM NAME: 1800MACR

PROGRAM ADDRESS BLOCKS:	170000	/ABSOLUTE/	SIZE=000000	(ABS)
	167400	/SYSIVR/	SIZE=000400	(ABS)
	170000	/SYSROM/	SIZE=000047	(ABS)
	153600	/SDBGWS/	SIZE=000051	(ABS)
	156600	/SDATAS/	SIZE=000063	(ABS)

PRIMARY TRANSFER ADDRESS: 170000

FIXED ENTRY POINTS:

\$MACROM	000011	\$MACVER	000040	\$DISKBUF	154000	\$DOSEXT	156000
\$DATAS	156600	\$LOGORV	156617	\$SECTOR	156620	\$TRACK	156621
\$DEVADR	156622	\$DRVNUM	156623	\$DRVTAB	156624	\$KEYCHAR	170055
\$KBDINI	170060	\$CHARLOD	170063	\$DSPINIT	170066	\$DISPLAY	170071
\$DISPDOS	170074	\$CBL	170077	\$DCURSES	170102	\$BLNKDE	170105
\$BLINK	170110	\$FNC60	170113	\$FNC61	170116	\$FNC62	170121
\$FNC63	170124	\$FNC64	170127	\$FNC65	170132	\$FNC66	170135
\$FNC67	170140	\$FNC68	170143	\$FNC69	170146	\$FNC6A	170151
\$FNC68	170154	\$FNC6C	170157	\$SEEK	170162	\$STEP	170165
\$READ	170170	\$WRITE	170173	\$ONLINE	170176	\$RESTORE	170201
\$DOSDRV	170204	\$DOIO	170207	\$BUFIO	170212	\$STATUS	170215
\$FDSTAT	170220	\$NS	000203	\$ES	000003	\$HD	000224
\$VA	000213	\$CK	000207	\$EOL	000022	\$BP	000007
\$OI	000200	\$H	000011	\$V	000013	\$EL	000015
\$EOF	000021	\$RU	000023	\$RD	000024	\$F	000033
\$HA	000211	\$NL	000215	\$HU	000223	\$O	000233
\$OS	000100	\$OW	000040	\$MACVRS	177777		

UNUSED LABELS:

KEYINF

1,		;	1800 MACRO ROM PROGRAM - MAIN MODULE	JUN 14, 1978	12:05	HSP/HJS
2,		;				
3,		;				
4,	000011	\$MACROM*	EQU	9		MACRO ROM VERSION NUMBER
5,	000040	\$MACVER*	EQU	1		MACRO ROM REVISION LETTER
6,		*				
7,		.	9	HJS	20 JUL 78	VERSION 9 FINAL RELEASE
8,		.	9,J	HJS	21 JUN 78	FIX TWO RESTART BUGS (STL & BETA)
9,		.	9,I	HJS	14 JUN 78	FIX BUGS & ADD MORE SPACE IN CODE
10,		.	9,H	HJS	10 MAY 78	CORRECT BUGS & SETUP CRC FOR RELEASE
11,		.	9,G	HJS	24 MAR 78	ADD RIM BUFFER TEST
12,		.	9,F	HJS	20 MAR 78	ALLOW ROM TYPE 2 (3800) CODE GENERATION
13,		.	9,E	HJS	16 MAR 78	ADD NEW VECTOR, DISKETTE LOCKUP CONDITION
14,		.	9,C	HJS	13 FEB 78	FIX 'UNI-KEY' UP & COMMENTS
15,		.	9,B	HJS	4 FEB 78	CORRECT AND INSERT 'UNIVERSAL KEYBOARD'
16,		.	9,A	HJS	30 JAN 78	DISPLAY AND POR FIXUPS
17,		;				
18,	000002	PN1800	EQU	2		1800 PROCESSOR NUMBER (FROM INFO)
19,		;				
20,			SNAPOPT	6		INSURE 6600 CODE AVAILABLE
21,			LIST	F		
22,		;				
23,	000000	ASCII	EQU	0		NON-ZERO -> ASCII KEYIN IN DEBUG
24,	000000	ORIGINI	EQU	0		NON-ZERO -> ORIGIN CAPABILITIES IN DEBUG
25,	000000	STACKP	EQU	0		NON-ZERO -> STACK PUSH/POP OP'S IN DEBUG
26,		;				
27,	000002	RAMROM	EQU	2		ZERO -> ROM IN RAM AT 0010000
28,		.				ONE -> ROM IN RAM AT 0120000
29,		.				TWO -> ROM IN ROM AT 0170000
30,		;				
31,	000000	ROMTYPE	EQU	0		ZERO -> 1800 STANDARD MACRO-ROM
32,		.				ONE -> APF SPECIAL VERSION
33,		.				TWO -> RIMDLL VERSION (3800)
34,		.				THREE-> CYNTHIA &/OR RIM BOOT
35,		;				
36,	000000	ROMCRC	EQU	0		SET THIS EQU TO THE CRC OF BYTES 0170000
37,		.				TO 0177774, STARTING FROM AN INITIAL -1
38,		.				PUT THE CRC IN BYTES 0177775 & 6 (MSB/LSB)
39,		;				
40,	000013	BL	EQU	11		NUMBER OF BOTTOM LINE FOR DOS FUNCTION 6
41,	000000	LC	EQU	0		NUMBER OF LEFT COLUMN FOR DOS FUNCTION 6
42,		;				
43,	000106	NECP	EQU	70		'NO ENTRY' CURSOR POSITION FOR DEBUG
44,	000052	OPCODEBP	EQU	052		BREAK POINT OPCODE

45.
46.
47.
48.
49.
50.
51.
52.
53.

*
* GET ALL THE DEFINITIONS

INC	1800/MAC	INSTRUCTION DEFINITIONS
INC	MDEF1800	MEMORY AND BIT DEFINITIONS
INC	ODEF1800	DISKETTE DEFINITIONS
LIST	=I	
INC	STRUCT	STRUCTURED ASSEMBLER MACROS (FOR RIMDLL)
LIST	I	


```

56,
57,
58,
59, 153600
60, 153600
61,
62, 153600
63, 153602
64, 153603
65, 153604
66, 153606
67,
68, 153607
69, 153611
70,
71,
72, 000012
73,
74,
75,
76,
77, 000003
78, 153613
79, 153651
80,
81,
82,
83,
84,
85,
86,
87,
88,
89,
90,
91,
92,
93,
94,
95,
96,
97, 154000
98,
99, 156000

```

```

*
. DEFINE THE DEBUG WORKING STORAGE LOCATIONS
.
SDBGWS   ORG      SDBGWS
          USE      SDBGWS

.
CURADR   SK       2      LAST MEMORY LOCATION ADDRESSED
CUROUT   SK       1      LAST CHARACTER WRITTEN TO DEVICE
CURSTA   SK       1      LAST STATUS FROM A DEVICE
MODVAL   SK       2      LAST 'MODIFY' VALUE GIVEN
KEYINF   SK       1      ASCII KEYIN MODE FLAG (0=>NORMAL)

.
OLDTOS   SKIP     2      OLD TOP OF STACK
OLDREGS  SKIP     2      POINTER TO ALPHA OR BETA REGISTERS

.
BPTLN    IFC      ORIGINI
          EQU      10     MAXIMUM NUMBER OF BREAKPOINTS
          XIF
          IFS
BPTLN    EQU      ORIGINI
          6              MAXIMUM NUMBER OF BREAKPOINTS
          XIF
BPTES    EQU      3      THERE ARE (3) BYTES PER ENTRY
BPTABL   SK       BPTLN*BPTES
BPTABE   END OF BREAKPOINT TABLE

*
OTABLN   IFS      ORIGINI
          EQU      6      MAXIMUM NUMBER OF ORIGINS
CURORG   SKIP     2      CURRENT ORIGIN VALUE
CUROSN   SKIP     1      CURRENT ORIGIN SELECT NUMBER
OTABL    SKIP     OTABLN*2
OPTABE   ORIGIN TABLE
          END OF ORIGIN TABLE
          XIF

*
. THE REST OF THE PAGE IS USED FOR DEBUG'S REGISTERS AND STACK
.
          IFGE      $,SDBGWS+(SESAB,AND,0177)
          ERR       DEBUG WORKING STORAGE AREA TOO LARGE!
          XIF

*
. PUT THE DATA AREA DEFINITIONS INTO THE ENTRY POINT TABLE
.
$DISKBUF* EQU      DOSBUF      DOS LFN-LF3
$DOSEXT*  EQU      DOSEXT      DOS EXTENSION AREA

```

```

100,
101,
102,
103, 156600
104, 156600
105,
106, 156600
107, 156600
108, 156603
109, 156605
110, 156616
111, 156607
112, 156617
113,
114,
115,
116, 156620
117, 156621
118,
119, 156622
120, 156623
121,
122, 156624
123,
124,
125,
126, 000007
127,
128,
129, 000044
130,
131,
132,
133, 156644
134, 156662
135, 156653
136,
137,
138,
139,
140,
141,

```

```

*
, DEFINE THE DISK WORKING STORAGE MEMORY LOCATIONS
*
$DATAS      ORG      DOSEXT+0600
              USE      $DATAS

$DATAS*     EQU      $              DEFINE CURRENT DATAPAGE
              SKIP    3              * NOT USED *
$SAVXA      SKIP    2              TEMP HOLD FOR (XA)
$SAVBC      SKIP    2              TEMP HOLD FOR (BC)
$DSKREGS    EQU     $+7            DISK DRIVER REGISTER SAVE AREA
              SKIP    8
$LOGDRV*    SKIP    1              CURRENT DOS LOGICAL DRIVE (INIT =1)
*
, THE FOLLOWING TWO VARIABLES ARE ORDER DEPENDENT
*
$SECTOR*    SKIP    1              CURRENT SECTOR (ORDER DEPENDENT)
$TRACK*     SKIP    1              CURRENT TRACK (ORDER DEPENDENT)
*
$DEVADR*    SKIP    1              FLOPPY PHYSICAL DEVICE ADDRESS
$DRVNUM*    SKIP    1              FLOPPY LOGICAL DRIVE (FODR0/FODR1)
*
$DRVTAB*
*
, CURRENT SECTOR/TRACK FOR EACH DOS LOGICAL DRIVE $SECTOR/$TRACK
*
$DRVTBLN    EQU     7              ,AND, MASK FOR MAX # DOS DRIVES
              RPT     $DRVTBLN+1    LOGICAL DRIVES 0-7
              SKIP    2
$DATAL      EQU     $=$DATAS      LENGTH OF VARIABLES TO BE INITIALIZED.
*
, DISPLAY DRIVER SCRIBBLE AREAS
*
ROWS        SKIP    7              "CHARLOD" WORKING AREA
DSPREGSA    EQU     $+7            DISPLAY INTERFACE REG SAVE AREA
              SKIP    8
*
, --- THE REST OF THE PAGE IS NOT USED ---
*
LIST        I
*
INC         1800REST

```

. 1800 MACRO ROM RESTART MODULE JUN 20, 1978 13:20 HSP/HJS

. PHYSICALLY LOCATE THE CODE FOR THE ROM MAKING PROGRAM

. IFEO RAMROM,0
. SET 010000
. LOC SYSROM
. IFEO ROMTYPE,2
. ERR RIM=DLL MODULE WILL CAUSE ASSEMBLY ERRORS
. XIF

. IFEO RAMROM,1
. SET 0120000
. LOC SYSROM
. IFEO ROMTYPE,2
. ERR RIM=DLL MODULE WILL CAUSE ASSEMBLY ERRORS
. XIF

. IFEO RAMROM,2
. SET 0170000
. XIF

. MACROM
. SRPOWER

JUMP POWERUP POWER=UP ENTRYPOINT BY JUMP!

. SRRSTRY
JUMP RESTART RESTART ENTRYPOINT BY JUMP!

. SRSYSMF
. SECOND FAULT DURING FAULT DETECTION!
. ASSUME ALL SYSTEM MEMORY & CONTROL BAD
. (SAVE BYTE & POINT INTO STACK AREA)
. (ANYWHERE WILL DO IN THE STACK)
. NOTE: D=VALUE HAPPENS TO MAKE THIS WORK
. USE STANDARD STACK AS A DEFAULT STACK
. ! RETURN ADDRESS FROM SUBROUTINE !
. TO INIT STL, BASING, & INTERRUPT VECTOR

. LD SESTACK>8
. LED

. SYSMOV DE
. DE SYSPAT
. JMP MTSETX

. SRMEMPE
CLICKR MEMORY PARITY ERROR CAUSES CLICK
JMP SVMEMP THEN JUMP TO THE RAM VECTOR

. SRCCLICKR
ACLATT 3 SET LOUDNESS TO MAXIMUM AND
ACJUMP CLICKROM GO TO CLICK ROM CODE

. SRSTPE
CLICKR SECTOR TABLE PARITY ERROR CAUSES CLICK
JMP SVSTPAR THEN JUMP TO THE RAM VECTOR

. ONEMSA EJMP 0 RAM SERVICE INTERRUPT DOES EI AND JUMP
. (USE WASTED SPACE)

. SRTMOUT JMP TIMEOUT I=DECODE PROCESSOR TIMEOUT ERROR
IFNE S+3,SRNEXTAL

3,F
4,F
5,F
6,F
7,F
8,F
9,F
10,F
11,F
12,F
13,F
14,F
15,F
16,F
17,F
18,F
19,F
20,F
21,F
22,F
23,F 170000
24,F
25,F
26,F 170000
27,F
28,F 170000 104 266 363
29,F
30,F 170003 104 002 364
31,F
32,F
33,F
34,F 170006 036 357
35,F 170010 343
36,F
37,F 170011 174 065
38,F 170013 046 274 036 361
39,F 170017 104 005 373
40,F
41,F 170022 111 153
42,F 170024 104 000 357
43,F
44,F 170027 253
45,F 170030 210 254 360
46,F
47,F 170033 111 153
48,F 170035 104 074 357
49,F
50,F 170040 111 050 000 000
51,F
52,F
53,F
54,F

55.F
56.F
57.F
58.F 170044 106 242 372
59.F 170047 066 014
60.F 170051 104 055 362

	ERR	ROM VECTOR ADDRESSING MIS-MATCH ERROR
	XIF	
TIMEOUT	CALL	(PUT HERE TO FILL THE SPACE)
	LL	GET ADDRESS OF MESSAGE
	JMP	(USING WASTED SPACE)
	SSTATE	
	TIMEOUTM	
	ERROR	

63,F									
64,F	170054	000			DC	0		* NOT USED *	
65,F									
66,F					IFNE	\$,0170055		* MAGIC NUMBER *	
67,F					ERR	JUMP VECTOR ADDRESSING ERROR !			
68,F					XIF				
69,F									
70,F									
71,F									
72,F									
73,F									
74,F	170055	104 143 373			SKEYCHAR*	JUMP	DOSF62N	DOSFUNC 6 SUB 2 WITHOUT BLINK 'DE'	
75,F	170060	104 117 362			SKBDSINI*	JUMP	KBDSPINI	INIT DISPLAY AND KEYBOARD CONTROL INFO	
76,F									
77,F	170063	104 207 362			SCHARLOD*	JUMP	CHARLOD	LOAD CHARACTER SET SUBROUTINE	
78,F	170066	104 004 374			SDSPINIT*	JUMP	DSPINIT	INITIALIZE THE DISPLAY	
79,F	170071	104 341 374			SDISPLAY*	JUMP	DISPLAY	1800/1500 DISPLAY ROUTINE	
80,F	170074	104 337 374			SDISPDOS*	JUMP	DISPDOS	DOS COMPATIBLE DISPLAY ROUTINE	
81,F									
82,F	170077	104 312 374			DSCBL*	JUMP	DSPCBL	COMPUTE A DISPLAY BUFFER LOCATION	
83,F	170102	104 237 374			DSCURSES*	JUMP	CURSES	SUSPEND THE CURSOR	
84,F	170105	104 221 373			DSBLNKDE*	JUMP	BLINKDE	BLINK THE CURSOR AT (DE)	
85,F	170110	104 244 373			DSBLINK*	JUMP	BLINK	BLINK THE CURSOR AT "SECPOS"	
86,F									
87,F	170113	104 077 373			DOSFNC60*	JUMP	DOSF60	DOS FUNCTION 6 SUBFUNCTION 0	
88,F	170116	104 125 373			DOSFNC61*	JUMP	DOSF61	DOS FUNCTION 6 SUBFUNCTION 1	
89,F	170121	104 140 373			DOSFNC62*	JUMP	DOSF62	DOS FUNCTION 6 SUBFUNCTION 2	
90,F	170124	104 171 373			DOSFNC63*	JUMP	DOSF63	DOS FUNCTION 6 SUBFUNCTION 3	
91,F	170127	104 177 373			DOSFNC64*	JUMP	DOSF64	DOS FUNCTION 6 SUBFUNCTION 4	
92,F	170132	104 204 373			DOSFNC65*	JUMP	DOSF65	DOS FUNCTION 6 SUBFUNCTION 5	
93,F	170135	104 211 373			DOSFNC66*	JUMP	DOSF66	DOS FUNCTION 6 SUBFUNCTION 6	
94,F	170140	104 345 373			DOSFNC67*	JUMP	DOSF67	DOS FUNCTION 6 SUBFUNCTION 7	
95,F	170143	104 046 374			DOSFNC68*	JUMP	DOSF68	DOS FUNCTION 6 SUBFUNCTION 8	
96,F	170146	104 105 374			DOSFNC69*	JUMP	DOSF69	DOS FUNCTION 6 SUBFUNCTION 9	
97,F	170151	104 144 374			DOSFNC6A*	JUMP	DOSF610	DOS FUNCTION 6 SUBFUNCTION 10	
98,F	170154	104 204 374			DOSFNC6B*	JUMP	DOSF611	DOS FUNCTION 6 SUBFUNCTION 11	
99,F	170157	104 216 374			DOSFNC6C*	JUMP	DOSF612	DOS FUNCTION 6 SUBFUNCTION 12 * NEW *	

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- JUMP VECTORS TO EXTERNALLY USABLE ROUTINES AND ADDRESSES OF USEABLE TABLES -

```

100,F
101,F
102,F
103,F
104,F
105,F 170162 104 201 365
106,F 170165 104 252 365
107,F 170170 104 071 366
108,F 170173 104 025 366
109,F 170176 104 373 365
110,F 170201 104 230 366
111,F 170204 104 310 366
112,F
113,F 170207 104 135 366
114,F 170212 104 117 366
115,F 170215 104 327 365
116,F 170220 143
117,F 170221 007
118,F 170222 300
119,F
120,F
121,F
122,F
123,F
124,F
125,F
126,F
127,F
128,F
129,F
130,F
131,F
132,F
133,F
134,F
135,F
136,F
137,F

```

```

+
      IFLE      ROMTYPE,1
.
. JUMP VECTOR TO EXTERNALLY USABLE DISKETTE ROUTINES
.
$SEEK*   JUMP      $$SEEK      SEEK TO THE TRACK IN (D)
$STEP*   JUMP      $$STEP      STEP IN OR OUT 1 TRACK
$READ*   JUMP      $$READ      READ A SECTOR
$WRITE*   JUMP      $$WRITE     WRITE A SECTOR
$ONLINE*  JUMP      $$ONLINE    SEE IF DRIVE IS ON LINE
$RESTORE* JUMP      $$RESTORE   SEEK TRACK 0
$DOSDRIV* JUMP      $$DOSDRIV   CONVERT DOS DRIVE TO MICRO BUS DRIVE
.
$DOIO*   JUMP      $$DOIO      - NOT USED BY DOS -
$BUFIO*   JUMP      $$BUFIO    - NOT USED BY DOS -
$STATUS*  JUMP      $$STATUS    - NOT USED BY DOS -
$FDSTAT*  FDSTAT    - NOT USED BY DOS - (INLINE CAUSE SMALL)
      RET
      NOP
      XIF
      (FILL TO 3 BYTE ENTRY)
*
      IFGE      ROMTYPE,2
.
. JUMP VECTOR TO ERROR ROUTINE, NO USEABLE DISKETTE ROUTINES
.
$SEEK*   JUMP      ROMVERR      SEEK TO THE TRACK IN (D)
$STEP*   JUMP      ROMVERR      STEP IN OR OUT 1 TRACK
$READ*   JUMP      ROMVERR      READ A SECTOR
$WRITE*   JUMP      ROMVERR      WRITE A SECTOR
$ONLINE*  JUMP      ROMVERR      SEE IF DRIVE IS ON LINE
$RESTORE* JUMP      ROMVERR      SEEK TRACK 0
$DOSDRIV* JUMP      ROMVERR      CONVERT DOS DRIVE TO MICRO BUS DRIVE
.
$DOIO*   JUMP      ROMVERR      - NOT USED BY DOS -
$BUFIO*   JUMP      ROMVERR      - NOT USED BY DOS -
$STATUS*  JUMP      ROMVERR      - NOT USED BY DOS -
$FDSTAT*  JUMP      ROMVERR      - NOT USED BY DOS -
      XIF

```

140,F					*			
141,F	170223				BEEPCODE			
142,F					.			
143,F					. AUDIO CHANNEL CODE FOR A BEEP			
144,F					.			
145,F	170223	260	002		ACST	2	SET AUDIO CHANNEL STATUS	
146,F	170225	240	377		ACLLC	0,255	BEEP FOR 208 MS	
147,F	170227	000	000	000 040 077	BEEP01	DC	00,00,00,32,63,63,63,32	1200 HZ TONE
148,F	170237	230	012		ACDSBNZ	0,BEEP01		
149,F					.			
150,F	170241	240	017		ACLLC	0,15	PAUSE FOR 150 MS	
151,F	170243	241	144		BEEP02	ACLLC	1,100	
152,F	170245	231	002		BEEP03	ACDSBNZ	1,BEEP03	
153,F	170247	230	006		ACDSBNZ	0,BEEP02		
154,F	170251	200			ACHALT		ZERO STATUS AND HALT	
155,F					*			
156,F					.			
157,F					. AUDIO CHANNEL CODE FOR A CLICK			
158,F					.			
159,F	170252	260	000		CLIKCODE	ACSI	0	SET AUDIO CHANNEL STATUS
160,F	170254	077	077	077 077 077	CLICKROM	DC	63,63,63,63,63	GO HIGH FOR 500 US
161,F	170261	000	000	000 000 000		DC	00,00,00,00,00	GO LOW FOR 500 US
162,F	170266	040				DC	32	BACK TO THE MIDDLE
163,F	170267	270			ACKIL			HALT WITHOUT CHANGING STATUS (FOR CLICKR)

```

166,F
167,F 170270
168,F
169,F
170,F
171,F 170270 104 331 361
172,F 170273 104 331 361
173,F 170276 104 370 361
174,F 170301 104 370 361
175,F 170304 104 000 362
176,F 170307 104 000 362
177,F 170312 104 010 362
178,F 170315 104 010 362
179,F 170320 104 020 362
180,F 170323 104 020 362
181,F 170326 104 030 362
182,F 170331 104 030 362
183,F 170334 104 040 360
184,F 170337 104 040 360
185,F 170342 104 050 362
186,F 170345 104 050 362
187,F 170350 104 045 367
188,F 170353 104 045 367
189,F 170356 104 040 362
190,F 170361 104 040 362
191,F 170364 104 321 361
192,F 170367 104 321 361
193,F
194,F
195,F
196,F 170372 007
197,F 170373 000 000
198,F 170375 007
199,F 170376 000 000
200,F
201,F 170400 252
202,F 170401 210 223 360
203,F 170404 000 000
204,F
205,F 170406 252
206,F 170407 210 252 360
207,F 170412 000 000
208,F
209,F 000124

```

*
VECTI

.
. VERSION 2, 2200 COMPATIBLE INTERRUPT VECTOR INITIALIZATION VALUES

```

JUMP MEMPAT MEMORY PARITY ERROR
JUMP MEMPAT
JUMP INPART INPUT PARITY ERROR
JUMP INPART
JUMP OUTPAT OUTPUT PARITY ERROR
JUMP OUTPAT
JUMP WVIOLT WRITE VIOLATION
JUMP WVIOLT
JUMP AVIOLT ACCESS VIOLATION
JUMP AVIOLT
JUMP IVIOLT INSTRUCTION VIOLATION
JUMP IVIOLT
JUMP ONEMSA ONE MILLISECOND
JUMP ONEMSA
JUMP SYSCAT SYSTEM CALL
JUMP SYSCAT
JUMP BRKPNR BREAK POINT
JUMP BRKPNR
JUMP UAINST UNASSIGNED INSTRUCTION
JUMP UAINST
JUMP STPART SECTOR TABLE PARITY ERROR
JUMP STPART

```

*
. ADDITIONAL VECTOR INITIALIZATION VALUES

```

RET SVDISKWS - DISK WAITS
DA 0
RET SVDISPWS - DISPLAY WAITS
DA 0
ACLATT 2 SVBEEP - BEEP (0151)
ACJUMP BEEPCODE
DA 0 PAD OUT TO 6 BYTES
ACLATT 2 SVCLIK - CLICK (0153)
ACJUMP CLIKCODE
DA 0 PAD OUT TO 6 BYTES

```

.
VECTIL EQU \$-VECTI

212,	F								
213,	F	170414							
214,	F								
215,	F								
216,	F								
217,	F								
218,	F								
219,	F								
220,	F	170414	052	105	124	040	124		
221,	F	170432	052	105	060	040	123		
222,	F								
223,	F	170453	052	105	071	040	123		
224,	F	170470	052	105	061	040	115		
225,	F	170505	052	105	062	040	111		
226,	F	170521	052	105	063	040	117		
227,	F	170533	040	120	101	122	111		
228,	F								
229,	F	170545	052	105	064	040	127		
230,	F	170561	052	105	065	040	101		
231,	F	170573	040	120	122	117	124		
232,	F								
233,	F	170606	052	105	066	203	225		
234,	F	170614	052	105	067	203	225		
235,	F	170622	052	105	070				
236,	F	170625	040	111	116	123	124		
237,	F	170641	040	105	122	122	117		
238,	F								
239,	F	170652	052	105	115	040	115		
240,	F								
241,	F								
242,	F								
243,	F								
244,	F								
245,	F								
246,	F								
247,	F								
248,	F								
249,	F								
250,	F								
251,	F	170674							
252,	F								
253,	F								
254,	F								
255,	F								
256,	F	170674	111	153					
257,	F	170676	353						
258,	F	170677	364						
259,	F	170700	026	254					
260,	F	170702	250						
261,	F	170703	310						
262,	F	170704	035						
263,	F	170705	021						

```

*
* ERRORM
*
* ERROR MESSAGES
*
*
* IFGE ROMTYPE,2
ROMVERM DC '*EV ROM VECTOR',SNS,*MERROR
XIF
TIMOUTM DC '*ET TIMEOUT',SNS,*MERROR
SYSPRM DC '*E0 SYSTEM RAM',SNS,*MERROR
*
* STPARM DC '*E9 SECTOR',SNS,*MPARIT
MEMPARM DC '*E1 MEMORY',SNS,*MPARIT
INPARM DC '*E2 INPUT',SNS,*MPARIT
OUTPARM DC '*E3 OUTPUT'
MPARIT DC ' PARITY',SNS,*MERROR
*
* WVIOLM DC '*E4 WRITE',SNS,*MPROTE
AVIOLM DC '*E5 ACCESS'
MPROTE DC ' PROTECT',SNS,*MERROR
*
* IVIOLM DC '*E6',SNS,*MINSTE
SYSCAM DC '*E7',SNS,*MINSTE
UAINTM DC '*E8'
MINSTE DC ' INSTRUCTION'
MERROR DC ' ERROR* ',SES (AT LEAST 1 BLANK AFTER THE *, NO EEOL)
*
* MERXMSG DC '*EM MEMORY TEST',SNS,*MERROR USED BY 1800MOVI IF FATAL
* ERROR HAS OCCURED WHEN SHOULDN'T
*
* IFNE ERRORM>8,$>8
ERR ERROR MESSAGE ADDRESSING PROBLEMS
XIF
*
* ONEMSA
*
* ONE MILLISECOND - RE-ENABLE INTERRUPTS AND JUMP TO LOGICAL LOCATION ZERO
*
* EJMP 0 (AT 0170037 TO SAVE SPACE)
*
* SYSPAT
*
* SYSTEM RAM FAILURE - MUST FINISH CLEARING MEMORY AND RE-INITING TABLES
* BEFORE CAN DISPLAY ERROR MESSAGE
*
*
* CLICKR MAKE NOISE THAT ERROR HAPPENED
LHD
LLE DE = SVMEMP + VECTIL
LC 256-VECTIL
XRA
LBA ZERO OUT THE REST OF SYSTEM RAM (167400)
DECP HL
BT (ASSUME CAN READ WHAT WRITTEN)

```

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- INTERRUPT VECTOR HANDLERS SET UP AT POR AND RESTART -

264,F
265,F 170706 106 117 362
266,F
267,F 170711 106 242 372
268,F 170714 066 032
269,F 170716 104 055 362
270,F
271,F
272,F
273,F
274,F 170721 106 242 372
275,F 170724 006 053
276,F 170726 104 336 361
277,F
278,F 170731 106 242 372
279,F 170734 006 070
280,F
281,F 170736 115 164 207
282,F 170741 047
283,F 170742 176 044 277
284,F 170745 117 015
285,F 170747 176 064 100
286,F 170752 111 047
287,F 170754 027
288,F 170755 117 035
289,F 170757 176 064 100
290,F 170762 111 027
291,F 170764 360
292,F 170765 104 055 362
293,F
294,F
295,F
296,F
297,F
298,F
299,F
300,F 170770 106 242 372
301,F 170773 066 105
302,F 170775 104 055 362
303,F
304,F 171000 106 242 372
305,F 171003 066 121
306,F 171005 104 055 362

```

*
      CALL      KBDSPINI      INIT KEY TRANS, DISPLAY MEMORY & FONT
*
      CALL      SSTATE      NOW SWITCH TO DEBUG STACK
      LL        SYSPRM      POINT TO THE MESSAGE
      JMP       ERROR        GO DISPLAY IT AND ENTER DEBUG
*
* THE REST OF THE TRAPS SIMPLY SAVE THE STATE OF THE MACHINE AND GIVE A MESSAGE
*
STPART  CALL      SSTATE      SECTOR TABLE PARITY ERROR
        LA       STPARM
        JUMP     MSTPAT      THE REST IS LIKE MEMORY PARITY ERROR
*
MEMPAT  CALL      SSTATE      MEMORY PARITY ERROR
        LA       MEMPAM     POINT L-REG TO THE MESSAGE
*
MSTPAT  DPL       HL,OLDTOS   SWAP THE TOP TWO STACK ENTRIES
        DL       DE,HL      SO THE DISPLAY WILL BE OF THE
        ND       -1-64      BAD MEMORY LOCATION WHEN DEBUG
        INCP     HL,2        IS FINALLY ENTERED
        ORL      64
        DL       BC,HL      PICK UP OLD TOS-1
        DS       DE,HL      STORE OLD TOS AT TOS-1
        DECP     HL,2
        ORL      64
        DS       BC,HL      STORE OLD TOS-1 AT TOS
        LLA      (NOTE: CODE TO INC & DEC STACK POINTER)
        JUMP     ERROR
*
ROMVERR IFGE      ROMTYPE,2  ATTEMPT MADE TO USE DELETED ROM VECTOR
        CALL     SSTATE
        LL       ROMVERM
        JUMP     ERROR
        XIF
*
INPART  CALL      SSTATE      5500 BUS INPUT PARITY ERROR
        LL       INPARM
        JUMP     ERROR
*
OUTPAT  CALL      SSTATE      5500 BUS OUTPUT PARITY ERROR
        LL       OUTPAM
        JUMP     ERROR

```

307,F					*				
308,F	171010	106	242	372	WVIOLT	CALL	SSTATE	(SECTOR TABLE) WRITE VIOLATION	
309,F	171013	066	145			LL	WVIOLM		
310,F	171015	104	055	362		JMP	ERROR		
311,F					*				
312,F	171020	106	242	372	AVIDLT	CALL	SSTATE	(SECTOR TABLE) ACCESS VIOLATION	
313,F	171023	066	161			LL	AVIDLM		
314,F	171025	104	055	362		JUMP	ERROR		
315,F					*				
316,F	171030	106	242	372	IVIOLT	CALL	SSTATE	PRIV'D INSTRUCTION VIOLATION	
317,F	171033	066	206			LL	IVIOLM		
318,F	171035	104	055	362		JUMP	ERROR		
319,F					*				
320,F	171040	106	242	372	UAINST	CALL	SSTATE	UN-IMPLEMENTED INSTRUCTION VIOLATION	
321,F	171043	066	222			LL	UAINTM		
322,F	171045	104	055	362		JUMP	ERROR		
323,F					*				
324,F	171050	106	242	372	SYSCAT	CALL	SSTATE	SYSTEM CALL VIOLATION	
325,F	171053	066	214			LL	SYSCAM		
326,F						JUMP	ERROR		
327,F					*				
328,F					.TIMOUT	CALL	SSTATE	I-DECODE PROCESSOR TIMEOUT ERROR	
329,F						LL	TIMOUTM	(MOVED TO VECTOR POINT 0170044)	
330,F						JUMP	ERROR		
331,F					*				
332,F	171055				ERROR				
333,F					*				
334,F					. ERROR DISPLAY				
335,F					*				
336,F	171055	056	361			LH	MEMPAM>8	DISPLAY ERROR MESSAGE	
337,F	171057	046	013			LE	BL	ON THE BOTTOM LINE	
338,F	171061	036	050			LD	40	STARTING ON COLUMN 40	
339,F						LB	SOW	WRITE BLANKS & DO NOT WAIT ON DISPLAY KEY	
340,F	171063	313				LBD		(SAVE A BYTE)	
341,F	171064	106	341	374		CALL	DISPLAY	DISPLAY THE MESSAGE	
342,F	171067	076	327			LX	CURADR>8		
343,F	171071	113	144	207		DPL	DE,OLDTOS	CAUSE TOP STACK ENTRY	
344,F	171074	104	120	367		JMP	BRKPNN	TO BE DISPLAYED	

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- KEYBOARD AND DISPLAY INITIALIZATION ROUTINE & VALUES -

```

347,F
348,F
349,F
350,F
351,F
352,F 171077 060 061 062 063 064
353,F 171107 010
354,F 171110 070 071
355,F 171112 030
356,F 171113 010
357,F 171114 015
358,F 171115 056
359,F 171116 040
360,F 000020
361,F
362,F
363,F
364,F
365,F
366,F
367,F 171117
368,F
369,F
370,F
371,F 171117 066 150 056 357
372,F 171123 250
373,F 171124 340
374,F 171125 330
375,F 171126 027
376,F
377,F 171127 310
378,F 171130 026 020
379,F 171132 066 077 056 362
380,F 171136 036 334
381,F 171140 340
382,F 171141 021
383,F 171142 066 077
384,F 171144 026 020
385,F 171146 021
386,F 171147 353
387,F 171150 364
388,F 171151 376
389,F 171152 174 015
390,F 171154 006 001
391,F 171156 026 137
392,F 171160 021
393,F
394,F 171161 106 004 374
395,F
396,F 171164 250
397,F 171165
398,F 171165 066 216 056 363

```

```

*
. UNIVERSAL KEYBOARD TRANSLATE TABLE (WILL WORK ON 3600 KEYBOARD AS WELL)
.
.
. 000 TO 017      020 TO 037 IN TABLE
. UNSHIFTED KEYS  SHIFTED KEYS
. 0-7 NUMBER PAD  0-7 NUMBER PAD
. <BSP>           * UNDEFINED *
. 8,9 NUMBER PAD  8,9 NUMBER PAD
. <CAN> = SHIFTED <CAN> = UNSHIFTED
. * UNDEFINED *   <BSP>
. <ENT>           <ENT>
. ". " NUMBER PAD ". " NUMBER PAD
. * UNDEFINED *   <0> (SP)

UTRANT  DC      '01234567'
        DC      BSP
        DC      '89'
        DC      CAN
        DC      BSP
        DC      ENTER
        DC      ". "
        DC      ". "
UTRANTL EQU      $-UTRANT

.
. IFNE      $+1>8,UTRANT>8
. ERR      TABLE IS NOT TO CROSS PAGE BOUNDARIES
. XIF

.
*
. KBDSPINI
.
. KEYBOARD DISPLAY INITIALIZATION ROUTINE - INIT'S THE WHOLE KBD/DSP SYSTEM
.
.
. HL      SEKBS2
. XRA
. LEA
. LDA
. DS      DE,HL
.
. LBA
. LC      UTRANTL
. HL      UTRANT
. LD      SEKTRAN>8
. LEA
. BT
. LL      UTRANT
. LC      UTRANTL
. BT
. LHD
. LLE
. LML
. INCP    DE
. LA      1
. LC      128-UTRANTL-UTRANTL-1
. BT

.
. CALL    DSPINIT
.
. XRA
.
. KBDSPILP HL      TRIANGLE

```

ZERO OUT THE KEYBOARD CONTROL BYTES
(SEKBS2 & SEKBS1)

INIT THE TRANSLATE TABLE
NO AUTO INCREMENT OR STOP
SIZE OF UTRANT
USE LOWER 040 TRANSLATE TABLE
TO INIT THE KEYBOARD TRANSLATE TABLE

AND AGAIN (FOR SECOND HALF)

* CODE FOR SPACE (040) IS ONE-TO ONE *
REST OF TABLE IS ONE TO ONE
DO IT THE SIMPLEST WAY
FOR THE REMAINDER

INIT THE DISPLAY POINTERS AND CONTROLS

INIT THE DISPLAY FONT CODES

FILL THE DISPLAY FONT MEMORY

399.F 171171 155
400.F 171172 004 001
401.F 171174 074 200
402.F 171176 110 165 362
403.F
404.F 171201 066 272 056 362
405.F 171205 016 000
406.F
407.F

LDCCF
AD 1
CP 0200
JFZ KBDSPILP

WITH TRIANGLES FIRST

HL SPLCS
LB CURSOR
JUMP CHARLOD

THEN LOAD THE CHARACTERS IN THE TABLE
STARTING WITH THE CURSOR CHARACTER
AND RETURN FROM THERE (SAVE 3 BYTES)

```

410,F
411,F 171207
412,F
413,F
414,F
415,F
416,F
417,F
418,F
419,F
420,F
421,F
422,F
423,F
424,F
425,F
426,F
427,F
428,F
429,F
430,F
431,F
432,F
433,F
434,F 171207 307
435,F 171210 054 200
436,F 171212 053
437,F
438,F 171213 046 005
439,F 171215 160 223 362
440,F
441,F 171220 310
442,F 171221 015
443,F
444,F 171222 307
445,F 171223 070
446,F 171224 066 244 056 335
447,F
448,F 171230 012
449,F 171231 327
450,F 171232 062 212
451,F 171234 372
452,F 171235 015
453,F 171236 176 074 253
454,F 171241 110 230 362
455,F
456,F 171244 060
457,F 171245 015
458,F 171246 174 024 001
459,F 171251 110 222 362
460,F
461,F 171254 070

```

```

*
CHARLOD
.
. ENTER: HL -> LIST OF CHARACTER FONTS TO BE LOADED
.
. THE BYTES MATCH SPECIFICATIONS FOR BYTES ON THE 5500 AND THE ROUTINE
. AUTOMATICALLY CONVERTS THEM INTO THE 1800 LODCF INSTRUCTION FORMAT.
. THIS IS A LIST OF GROUPS OF FIVE BYTES EACH SPECIFYING A COLUMN OF
. THE FONT MATRIX.
. THE FONT ENTRIES MUST HAVE THE MSBIT CLEAR ON THE FIRST OF THE FIVE
. COLUMN FONT BYTES.
. THE ASCII CODE THE FIVE BYTES REPRESENT MUST PRECEED THEM AND THE HIGH
. ORDER BIT MUST BE SET TO MARK WHICH ASCII CODE IS BEING SPECIFIED.
. IF THE FIRST ENTRY IN THE LIST DOES NOT HAVE AN ASCII CODE, THE VALUE
. IN THE B-REG IS USED.
. IF THE ASCII CHARACTER IS THE NEXT ONE IN SEQUENCE, IT MAY BE LEFT OUT.
. THE LIST IS TERMINATED BY A 0200.
. THEREFORE, THE ONLY WAY TO LOAD CODE 000 IS IF IT THE FIRST ENTRY IN THE LIST
. AND THE B-REG IS ZERO.
.
. EXIT: ALL REGISTERS USED EXCEPT X-REG
. ONE STACK POSITION IN ADDITION TO THE CALL USED
. ZERO CONDITION TRUE
.
CHRLOP    LAM          GET THE LIST ENTRY
          XR           0200    CLEAR HIGH BIT IF SET (OR SET IT)
          RTZ          STOP IF TERMINATOR (0200) REACHED
.
          LE           5       SET THE COLUMN COUNT
          JTS          CHRGO    START CONVERSION IF FONT BIT NOW SET
.
          LBA          ELSE SAVE ASCII CHAR IN THE B-REG
          INCP         HL      POINT HL TO THE NEXT COLUMN BYTE
.
CHRCOL    LAM          GET THE COLUMN FONT BYTE
CHRGD     PUSH        HL      SAVE THE TABLE POINTER
          HL           ROWS    POINT TO TEMP GENERATION AREA
.
CHRRW     SRC          SET CARRY FROM LSB
          LCM
          ACCC         SHIFT CARRY BIT IN
          LMC
          INCP         HL      POINT TO THE NEXT ROW
          CPL          ROWS+7
          JFZ          CHRRW   CONTINUE FOR 7 ROWS
.
          POP          HL      HL => NEXT COLUMN FONT BYTE
          INCP         HL
          SUE          1
          JFZ          CHRCOL  LOOP UNTIL 5 COLUMNS DONE
.
          PUSH        HL      SAVE THE TABLE POINTER

```

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- LOAD CHARACTER SET ROUTINE AND SPECIAL POR CHARACTER SET -

```

462,F 171255 066 244 056 335
463,F 171261 301
464,F 171262 155
465,F 171263 060
466,F 171264 111 004 001
467,F 171267 104 207 362
468,F
469,F
470,F 171272
471,F
472,F
473,F
474,F 171272 177 177 177 177 177
475,F 171277 240
476,F 171300 000 000 000 000 000
477,F 171305 252
478,F 171306 024 010 076 010 024
479,F 171313 255
480,F 171314 010 010 010 010 010
481,F 171321 272
482,F 171322 000 021 000 000 000
483,F 171327 260
484,F 171330 076 105 111 121 076
485,F 171335 021 041 177 001 001
486,F 171342 043 105 105 111 061
487,F 171347 102 101 111 125 042
488,F 171354 014 024 044 177 004
489,F 171361 162 121 121 121 116
490,F 171366 036 051 111 111 106
491,F 171373 100 107 110 120 140
492,F 171400 066 111 111 111 066
493,F 171405 061 111 111 112 074
494,F 171412 301
495,F 171413 077 110 110 110 077
496,F 171420 177 111 111 111 066
497,F 171425 076 101 101 101 042
498,F 171432 177 101 101 042 034
499,F 171437 177 111 111 111 101
500,F 171444 177 110 110 110 100
501,F 171451 076 101 101 111 056
502,F 171456 177 010 010 010 177
503,F 171463 000 101 177 101 000
504,F 171470 006 001 101 176 100
505,F 171475 177 010 024 042 101
506,F 171502 177 001 001 001 001
507,F 171507 177 040 030 040 177
508,F 171514 177 060 010 006 177
509,F 171521 076 101 101 101 076
510,F 171526 177 110 110 110 060
511,F 171533 076 101 105 102 075
512,F 171540 177 110 114 112 061
513,F 171545 061 111 111 111 106

```

```

HL      ROWS      LOAD THE 1800 CHARACTER
LAB
LOOCF
POP      HL      RESTORE THE TABLE POINTER
ADB      1      BUMP B-REG TO NEXT ASCII CHAR
JMP      CHRLOP   GO THROUGH THE LOOP AGAIN

*
*
SPLCS
*
* SPECIAL CHARACTER SET FOR POWER UP INITIALIZATION OF THE RAM DISPLAY
*
DC      0177,0177,0177,0177,0177 000 (CURSOR CHARACTER)
DC      ' '+0200
DC      0000,0000,0000,0000,0000 040 (SPACE - BLANK)
DC      ' '*+0200
DC      0024,0010,0076,0010,0024 *
DC      ' '=+0200
DC      0010,0010,0010,0010,0010 -
DC      ' ':+0200
DC      0000,0021,0000,0000,0000 :
DC      '0'+0200
DC      0076,0105,0111,0121,0076 0
DC      0021,0041,0177,0001,0001 1
DC      0043,0105,0105,0111,0061 2
DC      0102,0101,0111,0125,0042 3
DC      0014,0024,0044,0177,0004 4
DC      0162,0121,0121,0121,0116 5
DC      0036,0051,0111,0111,0106 6
DC      0100,0107,0110,0120,0140 7
DC      0066,0111,0111,0111,0066 8
DC      0061,0111,0111,0112,0074 9
DC      'A'+0200
DC      0077,0110,0110,0110,0077 A
DC      0177,0111,0111,0111,0066 B
DC      0076,0101,0101,0101,0042 C
DC      0177,0101,0101,0042,0034 D
DC      0177,0111,0111,0111,0101 E
DC      0177,0110,0110,0110,0100 F
DC      0076,0101,0101,0111,0056 G
DC      0177,0010,0010,0010,0177 H
DC      0000,0101,0177,0101,0000 I
DC      0006,0001,0101,0176,0100 J
DC      0177,0010,0024,0042,0101 K
DC      0177,0001,0001,0001,0001 L
DC      0177,0040,0030,0040,0177 M
DC      0177,0060,0010,0006,0177 N
DC      0076,0101,0101,0101,0076 O
DC      0177,0110,0110,0110,0060 P
DC      0076,0101,0105,0102,0075 Q
DC      0177,0110,0114,0112,0061 R
DC      0061,0111,0111,0111,0106 S

```

514,F	171552	100 100 177 100 100	DC	0100,0100,0177,0100,0100	T
515,F	171557	176 001 001 001 176	DC	0176,0001,0001,0001,0176	U
516,F	171564	160 014 003 014 160	DC	0160,0014,0003,0014,0160	V
517,F	171571	177 002 014 002 177	DC	0177,0002,0014,0002,0177	W
518,F	171576	143 024 010 024 143	DC	0143,0024,0010,0024,0143	X
519,F	171603	160 010 017 010 160	DC	0160,0010,0017,0010,0160	Y
520,F	171610	103 105 111 121 141	DC	0103,0105,0111,0121,0141	Z
521,F	171615	200	DC	0200	=END-OF-LIST=
522,F					
523,F	171616	037 017 007 003 001	TRIANGLE DC	037,017,007,003,001,000,000	TRIANGLE IN 1800 FORMAT
524,F					

527,F
528,F
529,F
530,F
531,F 171625 014
532,F 171626 034
533,F 000002
534,F
535,F
536,F 171627 054
537,F 171630 074
538,F 171631 114
539,F 171632 134
540,F 171633 154
541,F 171634 174
542,F 171635 214
543,F 171636 234
544,F 171637 254
545,F 171640 274
546,F 171641 314
547,F 171642 334
548,F 171643 354
549,F 000015
550,F
551,F
552,F 171644 015
553,F 171645 035
554,F 171646 055
555,F 171647 075
556,F 171650 115
557,F 171651 135
558,F 171652 155
559,F 171653 175
560,F 171654 215
561,F 171655 235
562,F 171656 255
563,F 171657 275
564,F 171660 315
565,F 171661 335
566,F 171662 355
567,F 171663 375
568,F
569,F 000037
570,F
571,F
572,F
573,F
574,F
575,F 171664 054
576,F 171665 055

*
* SECTOR TABLE INITIALIZATION VALUES

*
* SMST DC ENCODED STL VALUE ADDR CARD# CARD#
S=SMST<4+STAE+STWE 150K = 12K 1 = 32K 1
SMSTL DC S=SMST<4+STAE+STWE 160K = 12K 1 = 32K 1
EQU S=SMST

*
* UMST DC ENCODED STL VALUE ADDR CARD# CARD#
S=SMST<4+STAE+STWE 000K = 12K 1 = 32K 1
DC S=SMST<4+STAE+STWE 010K = 12K 2 = 32K 1
DC S=SMST<4+STAE+STWE 020K = 12K 2 = 32K 1
DC S=SMST<4+STAE+STWE 030K = 12K 2 = 32K 1
DC S=SMST<4+STAE+STWE 040K = 12K 3 = 32K 1
DC S=SMST<4+STAE+STWE 050K = 12K 3 = 32K 1
DC S=SMST<4+STAE+STWE 060K = 12K 3 = 32K 2
DC S=SMST<4+STAE+STWE 070K = 12K 4 = 32K 2
DC S=SMST<4+STAE+STWE 100K = 12K 4 = 32K 2
DC S=SMST<4+STAE+STWE 110K = 12K 4 = 32K 2
DC S=SMST<4+STAE+STWE 120K = 12K 5 = 32K 2
DC S=SMST<4+STAE+STWE 130K = 12K 5 = 32K 2
UMSTL DC S=SMST<4+STAE+STWE 140K = 12K 5 = 32K 2
EQU S=UMST

*
* HMST DC ENCODED STL VALUE CARD#
S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 3
DC S=HMST<4+STA16+STAE+STWE = 32K 4
DC S=HMST<4+STA16+STAE+STWE = 32K 4
DC S=HMST<4+STA16+STAE+STWE = 32K 4
DC S=HMST<4+STA16+STAE+STWE = 32K 4
DC S=HMST<4+STA16+STAE+STWE = 32K 4
DC S=HMST<4+STA16+STAE+STWE = 32K 4
DC S=HMST<4+STA16+STAE+STWE = 32K 4

*
* PISTL EQU S=SMST POWER INITIALIZE SECTOR TABLE LENGTH

*
* IFNE S>8,SMST>8
ERR SECTOR TABLE ADDRESSING PROBLEMS - PAGE BOUNDARY
XIF

*
* WRAPST DC 2<4+STAE+STWE SPECIAL MEMORY SIZE TEST SECTOR TABLE
DC 2<4+STA16+STAE+STWE

DATAPOINT CONFIDENTIAL INFORMATION - SEE PAGE 1

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- POWER UP AND RESTART -

Address	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

631,F
632,F 172002
633,F
634,F
635,F
636,F
637,F
638,F
639,F 172002 040
640,F 172003 106 242 372
641,F 172006 111 153
642,F
643,F 172010 076 357
644,F 172012 105 151
645,F 172014 044 003
646,F 172016 074 001
647,F 172020 150 133 367
648,F
649,F 172023 066 225 056 363
650,F 172027 026 322
651,F 172031 062 077
652,F 172033 015
653,F 172034 100 033 364
654,F
655,F 172037 106 045 373
656,F
657,F 172042 250
658,F 172043 165
659,F 172044 062 161
660,F 172046 113 161
661,F
662,F
663,F 172050 105 150
664,F 172052 044 001
665,F 172054 152 117 362
666,F
667,F 172057 106 020 367
668,F
669,F 172062 111 030
670,F
671,F
142,F
143,F
144,F
145,F
146,F
147,F
148,F

*
RESTART

. RESTART - INVOKED BY THE TRANSITION FROM BOTH 'RESTART' AND 'RUN' KEYS
. DOWN TO NOT BOTH DOWN. THE DIAGNOSTIC ROUTINE IS ENTERED
. IF THE 'DISPLAY' KEY IS DOWN DURING THE TRANSITION.
. .

DI		DISABLE INTERRUPTS
CALL	SSTATE	SAVE THE MACHINE STATE
CLICKR		FORCE AUDIO TO CLEAR THE CHANNEL
LX	SEKBS2>8	SET UP THE PAGE REGISTER
PL	A,SEKBS1	JUMP TO DEBUG IF THE "DISPLAY" KEY
ND	SEDSPKY+SEKBDKY	IS DOWN AND NOT "KEYBOARD" KEY
CP	SEDSPKY	
JTZ	DEBUGI	
HL	SMST	INSURE THAT THE STACK WILL NOT MOVE
LC	(SEDSPBF>8,AND,0360)+SMSTL	
STLOC		WHEN RE-SECTORING HAPPENS IN MTSETUP
INCP	HL	* DELAY HERE TO ALLOW KBD STATUS TO SETTLE
JFC	RESWAIT	(USES HL FOR APPROX. 40 MILLISEC DELAY)
CALL	SETUP	INIT SECTORING AND BASING
IFLE	ROMTYPE,1	
XRA		
SISTART		STOP ANY COMMING IN PROGRESS
SIMODOUT		STOP ANY TALKING IN PROGRESS
SIACUOUT		STOP ANY DIALING IN PROGRESS
XIF		
PL	A,SEKBS2	CLEAR THE SCREEN IF THE "F1" KEY
ND	SEFUNC1	IS UP, AND
CTZ	KBDSPINI	INIT FONTS AND KEYBOARD TRANSLATE TABLE
IFLE	ROMTYPE,1	
CALL	RESTDSK	INIT DISK VARS TO ALL ONES
XIF		
ALPHAL		INSURE THAT DOS(WHATEVER) ENTERED IN ALPHA
JMP	RESTIPL	THE REST IS IN "1800BOOT"
		OR ITS VARIATIONS (SAVE 3 BYTES)

.
. . . WARNING: THE FIRST BYTE OF THE NEXT MODULE MUST BE LABELED "RESTIPL"
. AS THERE IS A FALL-THROUGH INTO IT TO DO THE RESTART BOOT
. FROM THE PREVIOUS MODULE.
. .

IFNE ROMTYPE,1
INC 1800BOOT

3,G
4,G
5,G
6,G 172064
7,G
8,G
9,G
10,G
11,G
12,G
13,G
14,G
15,G
16,G
17,G
18,G
19,G
20,G
21,G
22,G
23,G
24,G
25,G
26,G
27,G
28,G
29,G
30,G
31,G
32,G
33,G
34,G
35,G
36,G
37,G
38,G
39,G
40,G
41,G
42,G
43,G
44,G 172064 250
45,G 172065 076 335
46,G
47,G 172067
48,G 172067 106 115 364
49,G
50,G
51,G
52,G
53,G 172072 106 102 373
54,G 172075 130 064 364

1800 DISKETTE IPL RESTART ROUTINE MAR 20, 1978 16:00 HSP/HJS
RESTIPL
RESTART IPL BLOCK LOAD ROUTINE
IFED ROMTYPE,3
IF THE KEYBOARD KEY IS DOWN AND THERE IS A RIM ON THE I/O BUS
THEN LOAD THE RIM-DLL CODE INTO LOW MEMORY AND GO TO IT, ELSE
XIF
SCAN THROUGH THE DRIVES WAITING FOR ONE ON WITH A DISK ON LINE,
CONTROLLER READY, AND THE DISK NOT BUSY. IF NOTHING
WILL LOAD FROM THE CURRENT DRIVE, CHECK THE KEYBOARD STATUS
FOR THE DISPLAY KEY BEING DEPRESSED.
IF NOT DOWN, RESTART FROM THE TOP ALL OVER AGAIN.
IF DOWN, SCAN TO THE NEXT DRIVE IN SEQUENCE UNTIL ALL DRIVES SCANNED.
IF ALL DRIVES SCANNED, RESTART FROM THE TOP ALL OVER AGAIN.
IFED ROMTYPE,3
CALL DOSF60N
JFS RESTNRIM KEYBOARD KEY DOWN?
JFP RESTNRIM DISPLAY KEY DOWN ALSO?
RIMLOOP MLA *RIMDLLS+RIMADR CHECK THE RIM'S TO SEE IF ANY THERE
EX ADR
IN
ORA
JFZ RIMBOOT THERE IS AT LEAST ONE AT ITS ADDRESS
INCP HL
LAM
ORA
JFZ RIMLOOP CONTINUE TILL ALL RIMS TRIED
RIMWAIT CALL DOSF60N WAIT FOR BOTH KEYS COMING UP
JTS RIMWAIT
JTP RIMWAIT
XIF
IFNE ROMTYPE,2
RESTNRIM XRA
LX \$DATAS>8 SET UP THE PAGE REGISTER
RESTLOOP CALL LOADIPL TRY TO LOAD THE IPL BLOCK ON DRIVE (A)
IFED ROMTYPE,3
CALL BOOTCYN TRY TO BOOT THE CYNTHIA DISK
XIF
IFNE ROMTYPE,2
CALL DOSF60N (X & B REGS NOT DESTROYED)
JFP RESTIPL 'DSP KEY' NOT DOWN, TRY DRIVE ZERO

55.G
56.G 172100 105 217
57.G 172102 004 001
58.G 172104 074 010
59.G 172106 140 067 364
60.G
61.G 172111 153
62.G 172112 104 064 364

PL A,SLOGDRV
AD 1 TRY THE NEXT LOGICAL DRIVE
CP \$DRVTBLN+1 IF NOT ALL TRIED
JTC RESTLOOP

EX CLICK
JMP RESTIPL IF ALL TRIED, THEN START ALL OVER AGAIN

MACRO-PROCESSOR SYSTEM MACRO RQM CODE - HSP/HJS - 78JUL20 11:44
- 1800 DISKETTE IPL RESTART ROUTINE -

63,	G				
64,	G	172115			
65,	G				
66,	G				
67,	G				
68,	G				
69,	G	172115	106	310	366
70,	G	172120	111	143	
71,	G	172122	106	373	365
72,	G	172125	043		
73,	G				
74,	G	172126	106	230	366
75,	G	172131	043		
76,	G				
77,	G	172132	036	000	
78,	G	172134	046	001	
79,	G	172136	363		
80,	G	172137	056	330	
81,	G	172141	106	071	366
82,	G	172144	153		
83,	G	172145	043		
84,	G				
85,	G	172146	047		
86,	G	172147	117	015	
87,	G	172151	111	047	
88,	G	172153	117	015	
89,	G	172155	062	254	
90,	G	172157	111	253	
91,	G	172161	062	015	
92,	G	172163	003		
93,	G				
94,	G	172164	307		
95,	G	172165	260		
96,	G	172166	013		
97,	G				
98,	G	172167	105	217	
99,	G	172171	370		
100,	G	172172	174	070	
101,	G	172174	026	373	
102,	G	172176	250		
103,	G	172177	310		
104,	G	172200	021		
105,	G	172201	060		
106,	G	172202	015		
107,	G	172203	070		
108,	G	172204	007		
109,	G				
110,	G				
111,	G				
112,	G				

```

*
* LOADIPL
*
*   LOAD THE DISK IPL BLOCK (TRACK 0 PHYSICAL SECTOR 1) FROM DRIVE (A-REG)
*   AND EXECUTE IF GOOD FORMAT (LSB,MSB,-LSB,-MSB,0)
*
*   CALL      $$DOSDRIV      SELECT THE LOGICAL DRIVE GIVEN IN A-REG
*   FDSCLR                                MASTER CLEAR ANY DRIVE IN USE ***
*   CALL      $$ONLINE      EXIT IF DRIVE NOT ON LINE
*   RTC                                AND READY TO GO
*
*   CALL      $$RESTORE      TRY TO RESTORE THAT DRIVE
*   RTC                                EXIT IF NOT ON LINE
*
*   LD        0              IPL BLOCK ON TRACK 0
*   LE        1              SECTOR 1
*   LLD                                HL => PAGE WHERE IPL IS TO BE READ
*   LH        $DISKBUF>8
*   CALL      $$READ      READ THE IPL BLOCK
*   EX        CLICK      CLICK EACH ATTEMPTED READ
*   RTC                                EXIT IF BAD READ
*
*   DL        DE,HL          ELSE GET THE STARTING ADDRESS
*   INCP      HL,2
*   DL        BC,HL          AND THE 1'S COMPLEMENT
*   INCP      HL,2
*   XREC      SEE IF BC IS THE 1'S
*   XRDB      COMPLEMENT OF DE
*   INCP      BC            IF BC IS -1 (SHOULD BE) WILL SET CARRY
*   RFC      EXIT IF BAD IPL FORMAT
*
*   LAM      THE 5TH BYTE MUST BE ZERO
*   ORA
*   RFZ      EXIT IF BAD IPL FORMAT
*
*   PL        A,$LOGDRV      ELSE STORE DRIVE NUMBER THERE
*   LMA
*   PUSH     DE            SAVE THE DESTINATION ADDRESS
*   LC        251          MOVE THE NEXT 251 BYTES
*   XRA      FROM THE DISK BUFFER TO
*   LBA      THE STARTING ADDRESS
*   BT
*   POP      HL            START EXECUTING THE IPL
*   INCP     HL            AFTER THE DRIVE NUMBER
*   PUSH     HL
*   RET
*
*   IFEQ     ROMTYPE,3
*   INC      1800BCYN
*   XIF

```

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- 1800 DISKETTE IPL RESTART ROUTINE -

113,G	*			
114,G		IFGE	ROMTYPE,2	
115,G	RIMBOOT	DI		
116,G		XRA		
117,G		LBA		
118,G		LC	RIMDLLNL	THE DLL CODE IS MOVED TO BOTTOM MEMORY
119,G		LEA		
120,G		LDA		
121,G		HL	RIMDLLS	FROM ITS COPY IN ROM
122,G		BT		
123,G		JMP	000000	AND THEN JUMPED TO, NEVER TO RETURN
124,G				
125,G	* RIMDLLS	EDU	\$	RIM DOWN LINE LOAD ROUTINE
126,G		LOC	0	
127,G		INC	RIMDLL	
128,G		LOC	*	
129,G		XIF		
149,G		XIF		
150,G				
151,G	*	IFEQ	ROMTYPE,1	
152,G		INC	1800BAPF	(ABOVE WARNING HOLDS FOR THIS MODULE TOO!)
153,G		XIF		
154,G				
155,G	*	IFEQ	ROMTYPE,0	
156,G		INC	1800DIAG	

3, H
4, H
5, H
6, H 172205
7, H
8, H
9, H
10, H
11, H
12, H
13, H
14, H
15, H
16, H
17, H
18, H
19, H
20, H
21, H
22, H
23, H
24, H
25, H
26, H
27, H
28, H
29, H
30, H
31, H
32, H 172205 106 112 370
33, H
34, H 172210 106 004 374
35, H
36, H 172213 106 020 367
37, H 172216 106 157 365
38, H 172221 224 213 377 207 022
39, H
40, H 172236 043
41, H 172237 074 010
42, H 172241 100 213 364
43, H
44, H 172244 106 310 366
45, H 172247 106 230 366
46, H 172252 140 213 364
47, H
48, H 172255 106 157 365
49, H 172260 224 124 105 123 124
50, H
51, H 172333 043
52, H 172334 150 031 365
53, H
54, H 172337 074 003

. 1800 FLEXIBLE DISKETTE DIAGNOSTIC JUNE 8, 1978 10:30 HSP/HJS
. FDIAG
. ENTERED FROM DEBUG WITH THE "12345H" COMMAND
. PERFORMS THE FOLLOWING TESTS:
. 0. VERIFY - MEDIA VERIFY (SEQUENTIAL VERIFICATION OF ALL SECTORS)
UNTIL A SECTOR CAN NOT BE READ.
. 1&2. SEEK12 - ALTERNATE CYLINDER SEEK FROM 1 TO 0 OR 2 TO 1 WITH READ-VERIFY.
. 3. FIND - ATTEMPTS TO FIND TRACK 38 ON A DOUBLE DENSITY DISKETTE.
USED TO CO-RELATE ACCURACY OF TRACK 0 FLAG.
CONTINUOUSLY READS ANY SECTOR.
. 4. ALIGN - POSITION TO TRACK 38 AND HOLD THE HEAD DOWN
UNTIL THE KEYBOARD KEY IS DEPRESSED. SINGLE (ANY) DENSITY
READS WILL BE ATTEMPTED BUT THE DATA WILL NOT BE DISPLAYED.
. NOTE: - COMMANDS 0, 1, 2, AND 3
WILL WAIT AFTER EACH OPERATION
IF THE DISPLAY KEY IS DEPRESSED.
THEY WILL EXECUTE CONTINUOUSLY
UNTIL THE KEYBOARD KEY IS DEPRESSED OR
UNTIL THE SELECTED DRIVE GOES OFF LINE.
. CALL C12345 ENTRY MUST BE '12345H'
. CALL DSPINIT INIT DISPLAY POINTERS
. FDIAGDR CALL RESTDSK INIT ALL DRIVE CONTROLS TO ALL ONES
CALL FGETRSP GET THE DRIVE NUMBER
DC \$HD,\$VA,-1,\$CK,\$EOL,'DRIVE: ',SES
. RTC MANUAL EXIT FROM DISK DIAGNOSTIC
CP 8 MUST BE 0-7
JFC FDIAGDR
. CALL \$\$DOSDRIV SELECT THE DRIVE
CALL \$\$RESTORE AND RESTORE IT.
JTC FDIAGDR CATCH DRIVE NOT ON LINE
. CALL FGETRSP GET THE TEST NUMBER
DC \$HD,'TEST 0:VFY, 1:T1=0, 2:T2=1, 3:FND, 4:ALN ',SES
. RTC <0 MANUAL EXIT FROM DISK DIAGNOSTIC
JTZ FAUTO 0 => MEDIA VERIFY
. CP 3 1 => SEEK BETWEEN TRACKS 1 AND 0


```

55, H 172341 140 013 365
56, H 172344 150 001 365
57, H
58, H 172347 074 004
59, H 172351 110 213 364
60, H
61, H
62, H 172354
63, H
64, H
65, H
66, H 172354 036 046
67, H 172356 046 000
68, H 172360 106 201 365
69, H
70, H 172363 016 000
71, H 172365 076 335
72, H 172367 106 135 366
73, H 172372 106 102 373
74, H 172375 120 363 364
75, H
76, H 172400 007
77, H
78, H 172401
79, H
80, H
81, H
82, H 172401 036 046
83, H 172403 046 000
84, H 172405 106 125 365
85, H 172410 104 001 365
86, H
87, H 172413
88, H
89, H
90, H
91, H 172413 340
92, H 172414 334
93, H 172415 106 125 365
94, H 172420 113 024 001
95, H 172423 106 125 365
96, H 172426 104 014 365

```

```

      JTC      FSEEK      2 => SEEK BETWEEN TRACKS 2 AND 1
      JTZ      FFIND      3 => FIND TRACK 38 & DD READ IT
      *
      CP      4
      JFZ      FDIAGDR      >4 TRY ANOTHER NUMBER
      JTZ      FALIGN      4 => SEEK TO TRACK 38
      *
      FALIGN
      *
      SEEK TO TRACK 38 AND CONTINUOUSLY READ IT
      *
      LD      38      STEP TO CORRECT ALIGNMENT TRACK
      LE      0      (ASSUMING TRACK ZERO FLAG CORRECT)
      CALL    $$SEEK
      *
      FALoop   LR      FREAD      READ (ACCEPT ANY DATA)
      LX      $DATAS>8      (FROM THE CORRECT PAGE IN MEMORY)
      CALL    $$DDIO
      CALL    DOSF60N      TEST THE KEYBOARD KEY
      JFS      FALoop      TO KNOW WHEN TO END
      *
      RET
      *
      FFIND
      *
      FIND TRACK 38 ON A DOUBLE DENSITY DISKETTE
      *
      LD      38
      LE      0      ANY SECTOR WILL DO
      CALL    FCHECKR      CHECK AND READ THE SECTOR
      JMP     FFIND      CONTINUE TILL STOPPED
      *
      FSEEK
      *
      ALTERNATE CYLINDER SEEK FROM 2 TO 1 OR FROM 1 TO 0 WITH READ VERIFY
      *
      FSEEK1   LEA      SAVE 2 OR 1 AS MARKER OF SEEK TYPE
      LDE      TRACK 2 OR 1 SECTOR 2 OR 1
      CALL    FCHECKR      CHECK AND READ A SECTOR
      SUD      1      TRACK 1 OR 0 SECTOR 2 OR 1
      CALL    FCHECKR      READ THE OTHER TRACK/SECTOR
      JUMP     FSEEK1

```

```

97,H
98,H 172431
99,H
100,H
101,H
102,H 172431 036 000
103,H
104,H 172433 006 002
105,H
106,H
107,H
108,H
109,H
110,H 172435 340
111,H 172436 106 125 365
112,H 172441 140 076 365
113,H
114,H 172444 304
115,H 172445 004 002
116,H 172447 074 033
117,H 172451 140 035 365
118,H
119,H 172454 024 033
120,H 172456 110 035 365
121,H
122,H 172461 153
123,H 172462 113 004 001
124,H 172465 113 074 115
125,H 172470 140 033 365
126,H
127,H 172473 104 031 365
128,H
129,H 172476 066 172 056 365
130,H 172502 046 013 036 062
131,H 172506 106 337 374
132,H 172511 070
133,H
134,H 172512 060
135,H 172513 076 327
136,H 172515 046 221 036 335
137,H 172521 113 146 200
138,H 172524 007

```

```

*
FAUTO
.
. MEDIA VERIFY - READ ALL SECTORS ON ALL TRACKS UNTIL FAULT
.
. LD 0 START AT TRACK 0
.
. FAUTO0 LA 2 START AT SECTOR 2
. BY STARTING AT 2 (AND ENDING AT SECTOR 25)
. 26-1 TRANSITION IN MIDDLE OF TRACK READ
. 25-2 TRANSITION ACROSS TRACKS READ
. THIS GIVES MAX TIME FOR STEP TO COMPLETE
.
. FAUTO1 LEA
. CALL FCHECKR CHECK AND READ A SECTOR
. JTC FAUTOER
.
. LAE INCREMENT THE SECTOR NUMBER
. AD 2
. CP MAXSECT+1 SEE IF .LT. 27
. JTC FAUTO1 DO NEXT SECTOR IF SO
. HERE A = 28 IF HALF WAY & 27 WHEN DONE
. SU MAXSECT+1 BACK TO BEGINNING OF TRACK (IF BACK TO 1)
. JFZ FAUTO1
.
. EX CLICK CLICK ONCE EACH TRACK (WENT BACK TO ZERO)
. ADD 1 INCREMENT THE TRACK NUMBER
. CPD MAXTRAK+1 DO NEXT TRACK IF NOT UP TO MAXIMUM
. JTC FAUTO0
.
. JMP FAUTO ELSE DO THE DISK ALL OVER AGAIN
*
FAUTOER HL FAUTOMSG
DE LC+50<8+8L
CALL DISPDOS
PUSH (CORRECT FOR POP)
*
FDIAGEX POP FORGET NORMAL RETURN (FROM FCHECKR)
LX CURADR>8
DE STRACK
DPS DE,CURADR
RET RETURN TO DSPCAD (RATHER THAN JUMP)

```

139,	H								
140,	H	172525							
141,	H								
142,	H								
143,	H								
144,	H	172525	106	102	373				
145,	H	172530	170	125	365				
146,	H	172533	160	112	365				
147,	H								
148,	H	172536	106	373	365				
149,	H	172541	140	112	365				
150,	H								
151,	H	172544	106	201	365				
152,	H								
153,	H								
154,	H	172547	056	320					
155,	H	172551	106	071	366				
156,	H	172554	003						
157,	H	172555	153						
158,	H	172556	007						
159,	H								
160,	H	172557	060						
161,	H	172560	106	337	374				
162,	H	172563	070						
163,	H	172564	106	303	375				
164,	H	172567	024	060					
165,	H	172571	007						
166,	H								
167,	H	172572	007	052	105	122	122		
168,	H								
157,									
158,									
159,									
160,									
161,									
162,									
163,									
164,									

```

*
FCHECKR
*
* READ A SECTOR (IF ABLE TO) ROUTINE
*
CALL    DOSF60N    TEST KBD/DSP KEYS (NO CURSOR TO BLINK)
JTP     FCHECKR    DSP KEY MEANS 'HOLD IT'
JTS     FDIAGEX    KBD KEY MEANS 'THAT'S ALL, FOLKS!'
*
CALL    $$ONLINE   SEE IF DISK IS STILL ON-LINE
JTC     FDIAGEX    IT ISN'T, JUST RETURN
*
CALL    $$SEEK     SEEK TO THE TRACK IN THE D-REGISTER
                     BECAUSE OF CALL RESTDSK WILL DO RESTORE
                     INITIALLY (AND ALWAYS)
LH      SEDSPBF>8  READ A SECTOR INTO THE DISPLAY
CALL    $$READ     BUFFER (MUST START ON A PAGE)
RFC
EX      CLICK      NOTE THAT ERROR WAS FOUND (OF SOME SORT)
RET
*
FGETRSP POP         HL          HL => MESSAGE
CALL    DISPDOS    DISPLAY IT
PUSH    HL          PUT RETURN ADDRESS BACK ON STACK
CALL    KEYCHAR    AND GET RESPONSE
SU      '0'        UNBIAS ASCII DIGIT
RET
*
FAUTOMSG DC         $BP, '!*ERR*', $ES
*
XIF
*
IFEQ    ROMTYPE, 2
INC     1800RINT
XIF
*
IFLE    ROMTYPE, 1
INC     1800DSKR

```

3,I
4,I
5,I
6,I 172601
7,I
8,I
9,I
10,I
11,I
12,I
13,I
14,I
15,I
16,I
17,I
18,I
19,I 172601 106 371 366
20,I 172604 105 221
21,I 172606 074 377
22,I 172610 110 220 365
23,I
24,I 172613 106 233 366
25,I
26,I
27,I 172616 043
28,I
29,I 172617 250
30,I 172620 223
31,I 172621 053
32,I
33,I 172622 320
34,I 172623 136 221
35,I 172625 006 040
36,I 172627 120 237 365
37,I
38,I 172632 250
39,I
40,I 172633 222
41,I
42,I 172634 320
43,I 172635 006 020
44,I
45,I 172637 106 252 365
46,I 172642 043
47,I
48,I 172643 062 024 001
49,I 172646 110 237 365
50,I
51,I 172651 007

. 1800 DISKETTE INTERFACE ROUTINES MAY 10, 1978 12:13 HSP/HJS
.\$\$SEEK
.
SEEK (MOVE HEAD) TO A TRACK
.
ENTRY: D = DESIRED TRACK NUMBER (0-76)
STRACK = CURRENT TRACK NUMBER (ASSUMED TO BE ACCURATE)
.
EXITS: ALL REGISTERS PRESERVED
STRACK = NEW CURRENT TRACK # (ASSUMING ENTRY VALUE ACCURATE)
(TC) = DRIVE OFF LINE OR RESTORE UNSUCCESSFUL
(FC) = SEEK COMPLETED
5 EXTRA STACK LEVELS USED (MAX)
.
CALL \$DSKSAVR SAVE ALL REGISTERS AND LOAD X
PL A,STRACK PICK UP LAST-KNOWN TRACK
CP =1 JUMP IF TRACK IS KNOWN
JFZ \$\$SEEKOK
.
CALL \$RESTORX ELSE RESTORE (DON'T SAVE REGS AGAIN)
NOTE: THIS CALL HAS CALL BACK TO SEEK
AT LABEL SEEKOUT
CATCH DRIVE OFF LINE OR BAD RESTORE
.
RTC
.
XRA ELSE SET THE TRACK TO ZERO
SUD (ACTUAL)-(DESIRED)
RTZ DISK IS ALREADY THERE
.
LCA
PS D,STRACK
LA FOMVOT
JFS \$\$SEEKLOP
.
XRA ELSE GET 2'S COMP OF NEG DIFFERENCE
.
\$\$SEEKOUT SUC - ENTER HERE FROM \$\$RESTORE -
5 EXTRA STACK LEVELS USED (MAX) & WAIT OVERHEAD
.
LCA
LA FOMVIN AND MOVE IN TOWARD THE CENTER
.
\$\$SEEKLOP CALL \$\$STEP
RTC STEP IN OR OUT BASED ON A-REG
EXIT IF DRIVE OFF LINE
.
SUC 1 ELSE DECREMENT THE DISTANCE TO GO
JFZ \$\$SEEKLOP KEEP STEPPING IF NOT ZERO
.
RET ELSE RESTORE REGISTERS AND RETURN

```

52,I
53,I 172652
54,I
55,I
56,I
57,I
58,I
59,I
60,I
61,I
62,I
63,I
64,I
65,I
66,I
67,I
68,I 172652 107 203
69,I 172654 116 206
70,I 172656 114 223
71,I
72,I 172660 111 260
73,I 172662 105 222
74,I 172664 064 220
75,I 172666 145
76,I
77,I 172667 106 102 357
78,I
79,I 172672 105 222
80,I
81,I 172674 111 145
82,I
83,I 172676 301
84,I 172677 044 001
85,I 172701 152 275 366
86,I 172704 140 322 365
87,I
88,I 172707 301
89,I 172710 044 040
90,I 172712 110 267 365
91,I
92,I 172715 301
93,I 172716 044 100
94,I 172720 054 100
95,I
96,I 172722 114 206
97,I 172724 105 203
98,I 172726 007

```

```

+
$$STEP
. STEP IN OR OUT ONE TRACK
.
. ENTRY: A      = OPERATION (FOMVIN/FOMVOUT)
.          X      = $DATAS>8
.          $DRVNUM = DRIVE NUMBER (FODR0/FODR1)
.          $DEVADR = MICRO-BUS ADDRESS
.
. EXITS: ALL REGISTERS PRESERVED
.         (TC)(FZ) = DRIVE OFF LINE
.         (FC)(TZ) = TRACK 0 DETECTED
.         (FC)(FZ) = TRACK 0 NOT DETECTED
.         1 EXTRA STACK LEVELS USED (MAX) & WAIT OVERHEAD
.
. PS      A,$$AVXA      SAVE THE A-REGISTER
. PS      B,$$AVBC+1    SAVE THE B-REGISTER
. PL      B,$DRVNUM     B = DRIVE SELECT (1 OR 2)
. ...     -1=FODR0-FODR1 ONLY THE BOTTOM 2 BITS ARE OK
. NDB
. ORAB
. PL      A,$DEVADR     GET THE MICRO-BUS ADDRESS
. OR      FCOMOD        'OR' IN THE MODE
. UBOUT
.
*
$$STEPWTL CALL      SVDISKWS      CALL THE "WAIT" ROUTINE
.
. PL      A,$DEVADR     GET THE BUS ADDRESS
. ...     OR      FCINST      INPUT STATUS (= 0)
.         URIN          WAIT FOR STEP TO COMPLETE
.
. LAB
. ND      FSONLN        CATCH DRIVE OFF LINE
. CTZ     TOFFTL        TURN OFF THE LIGHT AND SET
. JTC     $$STEPXIT     CARRY TRUE IF OFF LINE
.
. LAB
. ND      FSSTIP        WAIT IF STEP IN PROGRESS
. JFZ     $$STEPWTL
.
. LAB
. ND      FSTRK0        ELSE SEE IF TRACK 0 DETECTED
. XR      FSTRK0        EXIT FALSE CARRY TRUE ZERO IF SO
.
. $$STEPXIT PL      B,$$AVBC+1    RESTORE (B)
.           PL      A,$$AVXA      RESTORE (A)
.           RET

```

```

99,I
100,I 172727
101,I
102,I
103,I
104,I
105,I
106,I
107,I
108,I
109,I
110,I
111,I
112,I
113,I 172727 022 070
114,I 172731 076 335
115,I 172733 105 222
116,I
117,I 172735 111 145
118,I 172737 301
119,I 172740 044 003
120,I 172742 074 003
121,I 172744 152 102 357
122,I 172747 150 333 365
123,I
124,I 172752 106 275 366
125,I
126,I 172755 105 222
127,I 172757 064 220
128,I 172761 114 223
129,I 172763 145
130,I 172764 105 222
131,I
132,I 172766 111 145
133,I 172770 022 060
134,I 172772 007

```

```

*
**STATUS
. GET THE DISKETTE STATUS
.
. NOTE: THIS ROUTINE SHOULD NOT BE CALLED IF TRANSFERS ARE STILL
.       IN PROGRESS. (NORMALLY DOESN'T HAPPEN, ALL I/O INSIDE $SDOIO)
.
. ENTRY: $DEVADR = MICRO-BUS ADDRESS
.         $DRVNUM = DRIVE NUMBER (FODR0/FODR1)
.
. EXITS: B = DEVICE STATUS
.         1 EXTRA STACK LEVELS USED (MAX)
.
. PUSH      XA      SAVE THE XA REGISTER PAIR
. LX        $DATAS>8 SET UP THE PAGE REGISTER
. PL        A,$DEVADR SEE IF THE CONTROLLER IS IN ACTION
. OR        FCINST   BY CHECKING ITS STATUS
.
. UBIN      THIS CAUSES A WAIT ON LAST ACTION IF
. LAB       ACCIDENTLY ENTER HERE TO CHANGE DRIVES
. ND        FSONLN+FSTRIP IF BOTH ONLINE AND TRANSFER IN PROGRESS
. CP        FSONLN+FSTRIP ARE SET (SOMETHING GOING ON)
. CTZ       SVDISKWS   WAIT (CC NOT MODIFIED) AND
. JTZ       STATWAIT   LOOP TILL ONE GOES AWAY
.
. CALL      TOFFTL     THIS WILL CLEAR FSTRIP IF STILL SET
.
. PL        A,$DEVADR GET THE MICRO-BUS ADDRESS
. OR        FCOMOD     SET MODE TO SELECT DRIVE
. PL        B,$DRVNUM  B = DRIVE NUMBER (FODR0/FODR1)
. UBOUT     SELECT A DRIVE
. PL        A,$DEVADR GET THE MICRO-BUS ADDRESS
. OR        FCINST     INPUT STATUS
.
. UBIN      GET STATUS INTO (B)
. POP       XA         RESTORE THE XA REGISTER PAIR
. RET

```

```

135,I
136,I 172773
137,I
138,I
139,I
140,I
141,I
142,I
143,I
144,I
145,I
146,I
147,I
148,I 172773 022 070
149,I 172775 076 335
150,I 172777 116 206
151,I 173001 106 327 365
152,I 173004 301
153,I 173005 044 001
154,I 173007 152 275 366
155,I 173012 140 020 366
156,I
157,I 173015 111 044 010
158,I 173020 114 206
159,I 173022 022 060
160,I 173024 007

```

```

+
$$ONLINE
.
. CHECK FOR DRIVE ON-LINE AND READY
.
. ENTRY: (SAME AS FOR $$STATUS)
.
. EXITS: ALL REGISTERS PRESERVED
. (TC)(FZ) = DRIVE OFF LINE OR NOT READY
. (FC)(TZ) = DRIVE ON LINE AND WRITE ENABLED
. (FC)(FZ) = DRIVE ON LINE AND WRITE PROTECTED
. 3 EXTRA STACK LEVELS USED (MAX)
.
. PUSH XA SAVE (XA)
. LX $DATA$B
. PS B,$$AVBC+1 SAVE (B)
. CALL $$STATUS GET DISKETTE STATUS
. LAB IN THE A-REGISTER
. ND FSONLN (DRIVE MAY BE CHANGED BY $STATUS CALL)
. CTZ TOFFTL TURN OFF THE LIGHT IF NOT READY
. JTC ONLINXIT JUST EXIT IF NOT ON LINE
.
. ONLINXIT NDB FSPRO SET FALSE ZERO IF FILE PROTECTED
. PL B,$$AVBC+1 RESTORE (B)
. POP XA RESTORE (XA)
. RET

```

```

161,I
162,I 173025
163,I
164,I
165,I
166,I
167,I
168,I
169,I
170,I
171,I
172,I
173,I
174,I
175,I
176,I
177,I
178,I 000007
179,I 000006
180,I
181,I 173025 106 371 366
182,I 173030 066 000
183,I 173032 105 222
184,I
185,I 173034 016 060
186,I 173036 141
187,I 173037 043
188,I
189,I 173040 016 050
190,I 173042 141
191,I 173043 043
192,I
193,I 173044 174 134 212
194,I 173047 016 050
195,I 173051 106 135 366
196,I 173054 110 113 366
197,I
198,I 173057 105 211
199,I 173061 074 006
200,I 173063 053

```

```

+
$$WRITE
.
. WRITE A SECTOR (DOUBLE DENSITY)
.
. ENTRY: H = MSB OF MEMORY BUFFER
.         D = TRACK TO BE WRITTEN
.         E = SECTOR TO BE WRITTEN
.         A = WRITE/ WRITE=VERIFY CONTROL
.
. EXITS: ALL REGISTERS PRESERVED
.        STRACK = TRACK WRITTEN
.        SSECTOR = SECTOR WRITTEN
.        (TC)   = MISSING D.C. GAP OR CRC ERROR DURING VERIFY
.        (FC)   = DATA WRITTEN FROM THE PAGE POINTED TO BY H-REG
.        3 EXTRA STACK LEVELS USED (MAX)
.
DSKWV EQU 7 DISK WRITE/VERIFY
DSKWRT EQU 6 DISK WRITE = NO VERIFY
.
CALL $DSKSAVR SAVE ALL REGISTERS AND LOAD X
LL 0 INITIALIZE (L) TO 01TH BYTE
PL A,$DEVADR GET THE DEVICE ADDRESS
.
LB F0WPI LOAD WRITE PREAMBLE
FDDATA
RTC CATCH OPERATION IN PROGRESS
.
LB FDOTD LOAD BUFFER DOUBLE DENSITY
FDDATA
RTC CATCH OPERATION IN PROGRESS
.
DPLR DE,$DSKREGS-4 RESTORE THE TRACK/SECTOR
LB FFWRITE+FFDBL DO DOUBLE DENSITY WRITE
CALL $$DOIO
JFZ $DSKERR CATCH NO D.C. GAP OR SECTOR NOT FOUND
.
PL A,$DSKREGS-5 GET THE OPERATION CODE
CP DSKWRT JUST EXIT TRUE ZERO FALSE CARRY
RTZ IF NON-VERIFIED WRITE OPERATION

```


DATAPOINT CONFIDENTIAL INFORMATION - SEE PAGE 1

MACRO=PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- 1800 DISKETTE INTERFACE ROUTINES -

```

201,I
202,I
203,I
204,I 173064 006 030
205,I 173066 104 100 366
206,I
207,I 173071
208,I
209,I
210,I
211,I
212,I
213,I
214,I
215,I
216,I
217,I
218,I
219,I
220,I
221,I
222,I 173071 106 371 366
223,I 173074 066 000
224,I 173076 006 010
225,I
226,I 173100 106 133 366
227,I 173103 110 113 366
228,I
229,I 173106 310
230,I 173107 105 222
231,I 173111 141
232,I 173112 003
233,I
234,I 173113 006 200
235,I 173115 200
236,I 173116 007
237,I
238,I 173117
239,I
240,I
241,I
242,I
243,I
244,I
245,I
246,I
247,I
248,I
249,I
250,I
251,I
252,I

```

```

*
. DO "WRITE-VERIFY" - RE-READ & COMPUTE CRC)
.
.      LA      FDVRD      DO DOUBLE DENSITY READ/VERIFY
.      JMP     READDATA
*
. $SREAD
.
. READ A SECTOR (DOUBLE DENSITY)
.
.      ENTRY: H = MSB OF MEMORY BUFFER
.             D = TRACK TO BE READ
.             E = SECTOR TO BE READ
.
.      EXITS: ALL REGISTERS PRESERVED
.             $TRACK = TRACK READ
.             $SECTOR = SECTOR READ
.             (TC)   = MISSING D.C. GAP OR CRC ERROR DURING READ
.             (FC)   = DATA READ INTO THE PAGE POINTED TO BY THE H-REG
.             3 EXTRA STACK LEVELS USED (MAX)
.
.      CALL     $DSKSAVR      SAVE ALL REGISTERS AND LOAD X
.      LL       0             MAKE HL => START OF BUFFER PAGE
.      LA       FDIND        CAUSE INPUT OF DOUBLE DENSITY DATA
.
. READDATA CALL     $$DOIDDR      DO DOUBLE DENSITY READ
.          JFZ     $DSKERR      CATCH SOMETHING WRONG
.
.      LBA
.      PL       A,$DEVADR      GET THE DEVICE ADDRESS
.      FDDATA    GET THE DATA CHECKING CRC
.      RFC       EXIT IF DATA READ OR VERIFY OKAY
.
. $DSKERR LA       0200        SET CARRY TRUE
.      ADA
.      RET
*
. $SBUFIO
.
. PERFORM AN FDDATA OPERATION
.
.      ENTRY: B      = SUBFUNCTION NUMBER
.             C      = VALUE USED BY FDDATA (IF B = FDOUT)
.             HL     = LOCATION OF DATA AREA IN MEMORY (IF USED)
.             $DEVADR = MICRO-BUS ADDRESS
.
.      EXITS: X      = MSB OF $DATAS
.             A,C    = SCRATCHED
.             B      = CHANGED (LEAST SIGNIFICANT TWO BITS)
.             2 EXTRA STACK LEVELS USED (MAX)
.
.      NOTE: !! TO MAKE ROOM, MAY WANT TO TRIM THIS ROUTINE DOWN TO

```

253,	I				
254,	I				
255,	I				
256,	I	173117	076	335	
257,	I	173121	105	222	
258,	I	173123	174	070	
259,	I	173125	070		
260,	I	173126	141		
261,	I	173127	060		
262,	I	173130	174	060	
263,	I	173132	007		
264,	I				
265,	I				
266,	I				
267,	I				
268,	I				
269,	I				
270,	I				
271,	I				
272,	I				
273,	I				
274,	I				
275,	I				
276,	I				
277,	I				
278,	I				
279,	I	173133	016	010	
280,	I				
281,	I	173135	107	203	
282,	I	173137	126	204	
283,	I	173141	113	146	205
284,	I				
285,	I	173144	105	222	
286,	I	173146	124	223	
287,	I				
288,	I	173150	111	141	
289,	I	173152	140	221	366
290,	I				
291,	I	173155	106	102	357
292,	I	173160	105	222	
293,	I	173162	341		
294,	I	173163	111	145	
295,	I	173165	301		
296,	I	173166	314		
297,	I	173167	044	001	
298,	I	173171	150	221	366
299,	I				
300,	I	173174	143		
301,	I	173175	320		
302,	I	173176	044	002	
303,	I	173200	110	155	366
304,	I				

```

SIMPLY FDDATA INSTRUCTION AND RETURN.
I.E. ALL REGS MUST BE INITIALIZED AND NONE SAVED

LX          $DATAS>8
PL          A,$DEVADR
PUSH        DE
PUSH        HL
FDDATA
POP         HL
POP         DE
RET

*
DO A DISK READ/WRITE OPERATION (FDCMD)

ENTRY: X = $DATAS>8
        B = SUBFUNCTION TO BE PERFORMED
        D = TRACK NUMBER
        E = SECTOR NUMBER

EXITS:  ALL REGISTERS PRESERVED
        $TRACK = TRACK USED (IF VALID)
        $SECTOR = SECTOR USED (IF VALID)
        (TZ)   = OPERATION SUCCESSFUL
        (FZ)   = OPERATION NOT SUCCESSFUL
        1 EXTRA STACK LEVELS USED (MAX) & WAIT OVERHEAD

$DOIOER LB          FFREAD+FFDBL          DO DOUBLE DENSITY READ

$DOIO   PS          A,$SAVXA              SAVE (A)
        PS          C,$SAVXA+1            SAVE (C)
        DPS         DE,$SAVBC             SAVE (DE)

        PL          A,$DEVADR              GET MICRO-BUS ADDRESS
        PL          C,$DRVNUM              GET DRIVE NUMBER (FDDR0/FDDR1)

FDCMD    JTC        $DOIOERR              DO THE I/O OPERATION
                                              CATCH SOMETHING WRONG

$DOIOW   CALL        SVDISKWS              DO A WAIT
        PL          A,$DEVADR
        LEB                     SAVE THE B-REGISTER
        UBIN
        LAB                     A = STATUS
        LBE                     RESTORE THE B-REGISTER
        ND          FSONLN                CATCH DRIVE OFF LINE
        JTZ         $DOIOERR

FSTAT    LCA                     RETURN DE = SECTOR/TRACK USED
        ND          FRBUSY                A = STATUS
        JFZ         $DOIOW                WAIT IF BUSY

```

305,I 173203 113 146 220
306,I 173206 062 044 004
307,I
308,I 173211 113 144 205
309,I 173214 124 204
310,I 173216 105 203
311,I 173220 007
312,I
313,I
314,I
315,I 173221 111 143
316,I 173223 064 001
317,I 173225 104 211 366
318,I
319,I 173230
320,I
321,I
322,I
323,I
324,I
325,I
326,I
327,I
328,I
329,I
330,I
331,I 173230 106 371 366
332,I
333,I 173233 250
334,I 173234 107 221
335,I 173236 026 374
336,I 173240 106 233 365
337,I 173243 043
338,I
339,I 173244 026 117
340,I
341,I
342,I 173246 006 040
343,I 173250 106 252 365
344,I 173253 043
345,I 173254 053
346,I
347,I 173255 062 024 001
348,I 173260 100 246 366
349,I
350,I 173263 007

DPS	DE,\$\$SECTOR	ELSE SAVE THE SECTOR/TRACK USED
NDC	FRINDX	EXIT TRUE ZERO IF NOT 4 INDEXES
.		
\$DOIOXIT	DPL	DE,\$\$AVBC
	PL	C,\$\$AVXA+1
	PL	A,\$\$AVXA
	RET	RESTORE (A)
.		
. DO A STATUS CLEAR AND CAUSE RETURN FALSE ZERO		
.		
\$DOIOERR	FDSCLR	
	OR	1
	JMP	\$DOIOXIT
.		
.\$\$RESTORE		
.		
. SEEK TRACK 0 ON SELECTED DRIVE		
.		
ENTRY: \$DRVNUM = DRIVE NUMBER (FODR0/FODR1)		
.\$DEVADR = PHYSICAL DEVICE ADDRESS		
.		
EXITS: ALL REGISTERS PRESERVED		
(TC) = DRIVE OFF LINE OR TRACK 0 NOT FOUND		
(FC) = DISKETTE AT TRACK 0 AND \$TRACK = 0		
4 EXTRA STACK LEVELS USED (MAX) & WAIT OVERHEAD		
.		
CALL	\$DSKSAVR	SAVE THE REGISTERS AND SET UP X
.		
\$RESTORX	XRA	- ENTERED HERE FROM \$\$SEEK -
	PS	SET THE TRACK NUMBER TO 0
	LC	=4
	CALL	\$SEEKOUT
	RTC	CAUSE "SEEK" TO STEP IN 4 TRACKS
.		
	LC	CATCH DRIVE OFF LINE
	MAXTRAK+3	THEN STEP OUT UNTIL TRACK 0
OR THE MAXIMUM NUMBER OF TRACKS		
(+3 AS A SAFETY OF PAST TRACK 77)		
\$RESTORL	LA	STEP OUT
	CALL	
	RTC	
	RTZ	CATCH DRIVE OFF LINE
EXIT RESTORING REGS WHEN TRACK 0		
.		
SUC	1	ELSE DECREMENT TRACK COUNT
JFC	\$RESTORL	KEEP GOING IF NOT OUT OF RANGE
ELSE EXIT TRUE CARRY FALSE ZERO		
	RET	RESTORE REGISTERS AND EXIT

```

351,I
352,I 173264
353,I
354,I
355,I
356,I
357,I
358,I
359,I
360,I
361,I
362,I
363,I 173264 076 335
364,I 173266 146 222
365,I 173270 304
366,I 173271 064 220
367,I 173273 313
368,I 173274 145
369,I
370,I
371,I 173275
372,I
373,I
374,I
375,I
376,I
377,I
378,I
379,I
380,I
381,I
382,I
383,I 173275 105 222
384,I 173277 064 060
385,I 173301 016 177
386,I 173303 145
387,I 173304 006 201
388,I 173306 200
389,I 173307 007

```

```

*
TOFFTLDE
.
.   TURN OFF THE LIGHT IN THE SPECIFIED DRIVE
.
.   ENTRY: D = DRIVE SELECT CODE (1 OR 2)
.           E = DRIVE CONTROLLER NUMBER (0 THRU 3)
.
.   EXITS: X      = $DATAS PAGE
.            $DEVADR = DRIVE CONTROLLER NUMBER
.            A,B    = SCRATCHED
.
.   LX      $DEVADR>8
.   PS      E,$DEVADR      SET THE CONTROLLER ADDRESS
.   LAE     SELECT THE DRIVE IN (D)
.   OR      FCOMOD
.   LBD
.   UBOUT
.
.   FALLS THROUGH INTO "TOFFL"
*
TOFFTL
.
.   TURN OFF THE LIGHT AND RETURN TRUE CARRY FALSE ZERO
.   ALSO WILL CLEAR ALL INTERRUPTS PENDING AND
.   RESET THE MASTER READ/WRITE FLIP/FLOPS
.
.   ENTRY: X      = $DATAS PAGE
.            $DEVADR = DEVICE (DRIVE ALREADY SELECTED)
.
.   EXITS: LIGHT TURNED OFF, CLEARED ALL INTERRUPTS & RESET MASTER FLOPS
.           (TC)(FZ)
.
.   PL      A,$DEVADR
.   OR      FCLEAR
.   LB      FKLOFF+FKMAST
.   UBOUT
.   LA      0201
.   ADA
.   RET

```

```

390,I
391,I 173310
392,I
393,I
394,I
395,I
396,I
397,I
398,I
399,I
400,I
401,I
402,I
403,I
404,I
405,I 173310 106 371 366
406,I
407,I 173313 105 207
408,I 173315 044 007
409,I 173317 310
410,I
411,I 173320 105 217
412,I 173322 271
413,I 173323 053
414,I
415,I 173324 056 335
416,I 173326 200
417,I 173327 140 343 366
418,I
419,I 173332 044 016
420,I 173334 066 224
421,I 173336 017
422,I 173337 113 144 220
423,I 173342 027
424,I
425,I 173343 116 217
426,I 173345 066 224
427,I 173347 301
428,I 173350 200
429,I 173351 017
430,I 173352 047
431,I 173353 113 146 220
432,I
433,I 173356 111 032
434,I 173360 006 001
435,I 173362 014 000
436,I 173364 107 223
437,I 173366 116 222
438,I 173370 007

```

```

+
$DOSDRIV
.
. SWAP IN DOS LOGICAL DRIVE INFO.
.
. ENTRY: A = DOS LOGICAL DRIVE NUMBER (0-7)
.
. EXITS: ALL REGISTERS PRESERVED
. $LOGDRV = SELECTED LOGICAL DRIVE (0 THRU 7)
. $TRACK = CURRENT TRACK FOR THAT DRIVE
. $SECTOR = CURRENT SECTOR FOR THAT DRIVE
. $DRVNUM = DRIVE SELECT CODE (1 OR 2)
. $DEVADR = DRIVE MICRO-BUS ADDRESS (0, 1, 2, OR 3)
. 2 EXTRA STACK LEVELS USED (WAX)
.
CALL $DOSKSAVR SAVE THE REGISTERS AND SET UP X
.
PL A,$OSKREGS-7 GET THE A-REGISTER BACK
ND $DRVNUM LIMIT TO THE MAXIMUM DRIVE NUMBER
LBA SAVE THE DRIVE NUMBER IN B-REG
.
PL A,$LOGDRV GET CURRENT DOS LOGICAL DRIVE NUMBER
CPB EXIT RESTORING REGISTERS
RTZ IF SAME DRIVE AS LAST TIME
.
LH $DRVNUM SET UP THE H-REGISTER
ADA GENERATE INDEX INTO $DRVNUM
JTC $DOSDF0 CATCH NO CURRENT DRIVE
.
ND $DRVNUM<1 MAKE SURE $LOGDRV VALUE GOOD
LL $DRVNUM INDEX INTO THE $DRVNUM TABLE
INCP HL,A
DPL DE,$SECTOR GET CURRENT $SECTOR/$TRACK
DS DE,HL SAVE IT IN THE $DRVNUM ENTRY
.
$DOSDF0 PS B,$LOGDRV SAVE THE NEW LOGICAL DRIVE NUMBER
LL $DRVNUM INDEX INTO THE $DRVNUM
LAB
ADA
INCP HL,A (CARRY WILL BE CLEAR!! AFTER THIS)
DL DE,HL GET THE $SECTOR/$TRACK FOR NEW DRIVE
DPS DE,$SECTOR STORE IT INTO THE CURRENT $SECTOR/$TRACK
.
SRER
LA FODR0 DIVIDE IT BY TWO & LSBIT TO CARRY
AC FODR1=FODR1 A = RIGHT HAND DRIVE IF CARRY CLEAR
PS A,$DRVNUM A = LEFT HAND DRIVE IF CARRY SET
PS B,$DEVADR SAVE THE DRIVE SELECT
RET AND SAVE THE MICRO-BUS ADDRESS
RESTORE REGISTERS AND EXIT

```

439	I								
440	I	173371							
441	I								
442	I								
443	I								
444	I								
445	I	173371	051	216	335				
446	I	173374	055						
447	I	173375	022	060					
448	I	173377	022	060					
449	I	173401	051	011	367				
450	I	173404	022	070					
451	I	173406	076	335					
452	I	173410	007						
453	I								
454	I	173411							
455	I								
456	I								
457	I								
458	I	173411	066	216	056	335			
459	I	173415	111	055					
460	I	173417	007						
461	I								
462	I	173420							
463	I								
464	I								
465	I								
466	I	173420	006	377					
467	I	173422	066	200	056	335	370		
468	I	173427	335						
469	I	173430	046	201					
470	I	173432	026	043					
471	I	173434	250						
472	I	173435	310						
473	I	173436	021						
474	I	173437	007						
165									
166									
167									
168									
169		173440	000	000	000	000	000		
170									
171									
172									
173									
174									
175									
176									

```

*
*SDSKSAVR
*
* DISK ROUTINE REGISTER SAVE AND PAGE REGISTER SET UP
*   1 STACK LEVEL ADDITIONALLY CREATED
*
*       PUSH      $DSKREGS
*       REGS
*       POP       XA           GET RID OF "SDSKREGS" ADR ON THE STACK
*       POP       XA           XA = RETURN ADDRESS
*       PUSH      $DSKRESR     MAKE RETURNS AFTER THIS GO TO "SDSKRESR"
*       PUSH      XA           PUT RETURN ADR BACK ON THE STACK
*       LX        $DATAS>8     SET UP THE PAGE REGISTER
*       RET
*
*SDSKRESR
*
* RESTORE THE DISK ROUTINE REGISTERS AND EXIT
*
*       HL        $DSKREGS
*       REGL
*       RET
*
*RESTDSK
*
* SET ALL DISK CONTROL VARIABLES SO ALL DRIVES MUST BE INITIALIZED (RESTORED)
*
*       LA        =1           INITIALIZE ALL THE DISK VARIABLES
*       MSA       *$DATAS      TO ALL 1'S
*       LDH
*       LE        $DATAS+1
*       LC        $DATAL=1
*       XRA
*       LBA
*       BT
*       RET
*       XIF
*
*       IFEQ      ROMTYPE,0     (MAKE SPACE HERE SO LAST ROMS THE SAME)
*       RPT       05
*       DC        0
*       XIF
*       IFEQ      ROMTYPE,2     (MAKE SPACE HERE SO LAST ROMS THE SAME)
*       RPT       0330
*       DC        0
*       XIF
*
*       INC       18000BUG

```

3,J
4,J
5,J
6,J 173445
7,J
8,J
9,J
10,J 173445 106 242 372
11,J
12,J 173450 111 124 207
13,J 173453 174 047
14,J 173455 174 035
15,J 173457 066 211 056 327
16,J 173463 117 015
17,J 173465 176 074 251
18,J 173470 150 120 367
19,J
20,J 173473 307
21,J 173474 015
22,J 173475 274
23,J 173476 110 063 367
24,J
25,J 173501 307
26,J 173502 273
27,J 173503 110 063 367
28,J
29,J 173506 015
30,J 173507 307
31,J 173510 174 370
32,J 173512 174 027
33,J 173514 035
34,J 173515 006 377
35,J 173517 370
36,J 173520 113 146 200
37,J 173523 151
38,J 173524 104 133 367
39,J
40,J 173527
41,J
42,J
43,J
44,J 173527 040
45,J
46,J 173530 106 242 372

• 1800 DEBUG COMMAND INTERPRETER JUN 14, 1978 12:00 HSP/HJS
• BRKPNR
• BREAK POINT EXECUTION ENTRY
•
CALL SSTATE SAVE THE STATE OF THE MACHINE
LX CURADR>8 INIT THE PAGE REGISTER (BY SSTATE)
DPL BC,OLDTOS GET THE BREAK POINT ADDRESS
DL DE,BC
DECP DE
HL BPTABL=BPTES+1 SEARCH TABLE FOR ADDRESS
BRKPNL INCP HL,2 BUMP MEMORY POINTER TO NEXT ENTRY
CPL BPTABE STOP WHEN AT END OF TABLE
JTZ BRKPNN
•
LAM HL ELSE SEE IF ADDRESSES MATCH
INCP HL
CPE
JFZ BRKPNL
•
LAM HL
CPD
JFZ BRKPNL
•
INCP HL RESTORE CONTENTS IF MATCH FOUND
LAM HL
LMA DE
DS DE,BC UPDATE TOS ENTRY WITH BP LOCATION
DECP HL CLEAR THE BREAK POINT
LA -1
BRKPNN LMA
DPS DE,CURADR DISPLAY THE POINT AS CURRENT ADDRESS
EX BEEP MAKE SOME NOISE
JMP DSPCAD THEN GO TO DEBUG
*
DEBUG
• RETURN FROM DEBUG "CALL" ENTRY
•
DI MAKE SURE INTERRUPTS WON'T BOTHER ME
• INTERRUPTS MAY BE ON AFTER THE 'CALL'
CALL SSTATE SAVE THE MACHINE STATE

47 J
48 J 173533
49 J
50 J
51 J
52 J
53 J
54 J
55 J

*
DEBUGI
*
* ENTRY FROM "RESTART" INVOCATION
*
KEYINS IFS ASCII
XRA CLEAR ASCII KEYIN MODE
PS A,KEYINF SET ASCII KEYIN MODE FLAG TO A=REG
XIF


```

58,J
59,J 173533
60,J
61,J
62,J
63,J 173533 076 327
64,J
65,J
66,J
67,J
68,J
69,J
70,J
71,J
72,J
73,J
74,J
75,J
76,J
77,J
78,J
79,J
80,J 173535 046 010
81,J 173537 111 124 200
82,J 173542 062 070
83,J 173544 106 154 372
84,J 173547 060
85,J 173550 070
86,J 173551 016 040
87,J 173553 327
88,J 173554 046 011
89,J 173556 036 111
90,J 173560 106 312 374
91,J 173563 371
92,J 173564 015
93,J 173565 111 027
94,J 173567 066 003
95,J 173571 106 156 372
96,J 173574 060
97,J 173575 111 047
98,J 173577 046 012
99,J 173601 106 154 372
100,J 173604 051 133 367

```

```

*
DSPCAD
*
* PUT UP DEBUG "CURRENT ADDRESS" DISPLAY
*
LX          CURADR>8      SET UP THE PAGE REGISTER (SAFTEY)
IFS         ORIGINI
DPL         DE,CURORG     DE = CURRENT ORIGIN
LAE
ORD         SET ZERO CONDX IF ORIGIN EQUALS ZERO
LE          BL=4          ERASE 4TH TO BOTTOM LINE
LD          73            IF ORIGIN IS ZERO
CTZ         D0SF69        (ERASE TO END OF LINE)
DPL         DE,CURORG     DE = CURRENT ORIGIN
DPL         BC,CURADR     BC => CURRENTLY ADDRESSED SLOT
SUEC        COMPUTE THE PHYSICAL ADDRESS
SB08
LAE
ORD
LE          BL=4          DISPLAY THE LOGICAL ADR ON 4TH TO BOTTOM
CFZ         DSPOC6        LINE IF CURRENT ORIGIN IS NOT ZERO
XIF
LE          BL=3          DISPLAY THE PHYSICAL ADDRESS ON
DPL         BC,CURADR     THE 3RD TO THE BOTTOM LINE
PUSH        BC            SAVE THE PHYSICAL ADDRESS
CALL        DSPOC6
POP         HL            RESTORE THE ADDRESS
PUSH        HL            LEAVE IT ON THE STACK
LB          040
LCM
LE          BL=2
LD          73
CALL        DSPCBL        (WRITE CHARACTERS IN BC-REGS)
LMB
INCP        HL
DS          BC,HL         PUT OUT BLANK, CHARACTER, BLANK
LL          3             DISPLAY THE CHARACTER VALUE
CALL        DSPOCR        (B CLOSE ENOUGH TO 0 FOR DSPOCR OF 3)
POP         HL            RESTORE THE ADDRESS OF THE ITEM
DL          BC,HL         DISPLAY THE 6 DIGIT CONTENTS
LE          BL=1          ON THE NEXT TO BOTTOM LINE
CALL        DSPOC6
PUSH        DSPCAD

```

```

101,J
102,J 173607
103,J
104,J
105,J
106,J 173607 046 013 036 106
107,J 173613 026 000
108,J 173615 312
109,J 173616 062 070
110,J 173620 106 303 375
111,J 173623 062 060
112,J 173625 076 327
113,J
114,J
115,J
116,J
117,J
118,J
119,J
120,J
121,J
122,J 173627 074 015
123,J 173631 150 247 370
124,J 173634 074 030
125,J 173636 150 207 367
126,J 173641 074 010
127,J 173643 150 131 370
128,J 173646 074 077
129,J 173650 150 071 372
130,J 173653 074 056
131,J 173655 150 340 370
132,J 173660 074 136
133,J 173662 150 324 370
134,J 173665 074 043
135,J 173667 150 066 371
136,J 173672 074 101
137,J 173674 140 150 370
138,J 173677 074 173
139,J 173701 100 150 370
140,J 173704 066 224 056 367
141,J 173710 074 133
142,J 173712 140 326 367
143,J 173715 066 040 056 367
144,J 173721 074 141
145,J 173723 140 150 370

```

```

*
GETCMD
. GET THE NEXT COMMAND FROM THE KEYBOARD
.
DE      NECP<8+BL      GET THE COMMAND LINE CURSOR POSITION
LC      0              ZERO THE COMMAND VALUE ACCUMULATOR
LBC
GETCML  PUSH          BC      SAVE THE COMMAND VALUE SO FAR
CALL    KEYCHAR        GET CHAR FROM KEYBOARD FLASHING CURSOR
POP      BC            RESTORE THE COMMAND VALUE SO FAR
LX      CURADR>8       SET UP THE PAGE REGISTER (SAFTEY)
.
IFS      ASCII
PL      L,KEYINF       CATCH BEING IN ASCII KEYIN MODE
ORLL
JFZ     KEYINA          ENTER ASCII CHARACTER IF SO
XIF
*
. CHECK FOR SPECIAL COMMAND CHARACTERS
.
CP      ENTER          DECODE THE COMMAND
JTZ     NEWADR
CP      CAN            CANCEL COMMAND IF CANCEL KEY
JTZ     GETCMD
CP      BSP           BACKSPACE ONE IF BACKSPACE KEY
JTZ     BACKSP
CP      '?'           CATCH IDENTIFICATION INQUIRY
JTZ     IDENT
CP      '!'           MODIFY AND INCREMENT
JTZ     MODINC
CP      'A'           MODIFY AND INCREMENT USING LAST VALUE
JTZ     MODAGN
CP      '##'          # - CLEAR ALL BREAK POINTS
JTZ     BPCLR         RESTORING VALUES
CP      'A'           TRY TO ACCUMULATE OCTALS IF NOT LETTERS
JTC     GETDIG
CP      ' ' ' +1      (LOWER CASE 'Z')
JFC     GETDIG
HL      = 'A' < 1 + CMDTS  HL => COMMAND LETTER TABLE
CP      'Z' +1        BASED ON SHIFT CASE
JTC     DOCMDL
HL      = ' ' < 1 + CMDTNS (LOWER CASE 'A')
CP      ' '
JTC     GETDIG

```

146,J
147,J 173726
148,J
149,J
150,J
151,J
152,J 173726 200
153,J 173727 017
154,J 173730 057
155,J 173731 070
156,J 173732 012
157,J 173733 115 164 211
158,J 173736 113 074 106
159,J 173741 007
160,J
161,J 173742
162,J
163,J
164,J
165,J
166,J 173742 200 371
167,J
168,J
169,J
170,J
171,J 173744 004 371
172,J 173746 360 371
173,J 173750 257 370
174,J 173752 326 371
175,J
176,J 173754 200 371
177,J 173756 200 371
178,J
179,J
180,J
181,J
182,J
183,J
184,J 173760 205 364
185,J
186,J
187,J
188,J
189,J
190,J
191,J
192,J
193,J
194,J
195,J 173762 211 370
196,J 173764 373 371
197,J

*
DOCMDL

. INDEX INTO THE ADDRESS TABLE POINTED TO BY (HL)
. AND JUMP OFF TO THE INDICATED ADDRESS.
.

ADA		INDEX INTO THE GIVEN TABLE
INCP	HL,A	BY THE COMMAND LETTER
DL	HL,HL	SET UP TO JUMP TO THE COMMAND
PUSH	HL	
SRC		RESTORE THE COMMAND LETTER
DPL	HL,OLDREGS	HL => REGISTER STORAGE
CPD	NECP	SET TRUE ZERO IF NO DIGITS ENTERED
RET		

*
CMDTNS

. UNSHIFTED COMMAND ROUTINE ADDRESS TABLE
.

IFNE	ROMTYPE,3	
DA	ADRDEV	A - ADDRESS THE GIVEN OR LAST I/O DEVICE
XIF		
IFEQ	ROMTYPE,3	
DA	GETCME	A - * NOT USED * (5500 I/O BUS)
XIF		
DA	BPSET	B - SET A BREAK POINT AT GIVEN OR CURADR
DA	CALL	C - CALL THE GIVEN OR CURRENT ADDRESS
DA	DECADR	D - DECREMENT THE CURRENT ADDRESS
DA	EXECUT	E - CONTINUE EXECUTION
IFNE	ROMTYPE,3	
DA	ADRDEV	F - FETCH NEXT DATA BYTE FROM CURRENT I/O
DA	ADRDEV	G - GOTO DATA MODE IN THE CURRENT I/O
XIF		
IFEQ	ROMTYPE,3	
DA	GETCME	F - * NOT USED * (5500 I/O BUS)
DA	GETCME	G - * NOT USED * (5500 I/O BUS)
XIF		
IFEQ	ROMTYPE,0	
DA	FDIAG	H - HARDWARE FLOPPY DIAGNOSTIC
XIF		
IFEQ	ROMTYPE,1	
DA	APFDMP	H - APP MEMORY DUMP
XIF		
IFEQ	ROMTYPE,2	
DA	TSTRIM	H - EXTERNAL RIM BUFFER TEST
XIF		
IFEQ	ROMTYPE,3	
DA	GETCME	H - * NOT USED * (HARDWARE DIAGNOSTIC)
XIF		
DA	INCADR	I - INCREMENT THE CURRENT ADDRESS
DA	JUMP	J - JUMP TO THE GIVEN OR CURRENT ADDRESS
IFS	ASCII	

198,J		DA	KEYINM	K - SET ASCII KEYIN MODE
199,J		XIF		
200,J		IFC	ASCII	
201,J	173766 125 370	DA	GETCME	K - * NOT USED * (ASCII KEYIN)
202,J		XIF		
203,J	173770 233 370	DA	LINK	L - LINK TO ADR POINTED TO BY CURADR
204,J	173772 343 370	DA	MODIFY	M - MODIFY THE CURRENT ADDRESS CONTENTS
205,J		IFS	ORIGINI	
206,J		DA	NOSADR	N - SET NON-OFFSET ADDRESS
207,J		DA	ORIGIN	O - CLEAR OR SELECT ORIGIN
208,J		XIF		
209,J		IFC	ORIGINI	
210,J	173774 125 370	DA	GETCME	N - * NOT USED * (ORIGIN OFFSET)
211,J	173776 125 370	DA	GETCME	O - * NOT USED * (ORIGIN SELECT)
212,J		XIF		
213,J		IFNE	ROMTYPE,3	
214,J	174000 133 371	DA	BASSET	P - DISPLAY BASE REGISTER OR LOAD WITH BC = 0100000
215,J				Q - LOAD THE SECTOR TABLE
216,J	174002 160 371	DA	STLOAD	
217,J		XIF		
218,J		IFEQ	ROMTYPE,3	
219,J		DA	GETCME	P - * NOT USED * (BASE REGISTER)
220,J		DA	GETCME	Q - * NOT USED * (SECTOR TABLE)
221,J		XIF		
222,J	174004 120 371	DA	REGMOD	R - SWITCH ALPHA/BETA REGISTER MODE
223,J	174006 357 370	DA	STACKD	S - DISPLAY THE SPECIFIED STACK ITEM
224,J	174010 360 375	DA	TSTMEN	T - START MEMORY TEST ('12345T')
225,J	174012 314 371	DA	EXEUSR	U - USER MODE EXECUTE
226,J		IFNE	ROMTYPE,3	
227,J	174014 273 371	DA	OUTXC4	V - EX COM4 TO LAST I/O DEVICE
228,J	174016 247 371	DA	OUTXWT	W - EX WRITE TO LAST I/O DEVICE
229,J	174020 254 371	DA	OUTXC1	X - EX COM1 TO LAST I/O DEVICE
230,J	174022 261 371	DA	OUTXC2	Y - EX COM2 TO LAST I/O DEVICE
231,J	174024 266 371	DA	OUTXC3	Z - EX COM3 TO LAST I/O DEVICE
232,J		XIF		
233,J		IFEQ	ROMTYPE,3	
234,J		DA	GETCME	V - * NOT USED * (5500 I/O BUS)
235,J		DA	GETCME	W - * NOT USED * (5500 I/O BUS)
236,J		DA	GETCME	X - * NOT USED * (5500 I/O BUS)
237,J		DA	GETCME	Y - * NOT USED * (5500 I/O BUS)
238,J		DA	GETCME	Z - * NOT USED * (5500 I/O BUS)
239,J		XIF		

240,J
241,J 174026
242,J
243,J
244,J
245,J 174026 310 370
246,J 174030 303 370
247,J 174032 302 370
248,J 174034 305 370
249,J 174036 304 370
250,J 174040 301 370
251,J 174042 125 370
252,J 174044 307 370
253,J 174046 321 371
254,J 174050 037 372
255,J 174052 125 370
256,J 174054 306 370
257,J 174056 125 370
258,J 174060 125 370
259,J
260,J
261,J
262,J
263,J 174062 125 370
264,J
265,J
266,J 174064 133 371
267,J
268,J
269,J
270,J
271,J 174066 125 370
272,J
273,J
274,J
275,J
276,J
277,J 174070 125 370
278,J 174072 125 370
279,J
280,J 174074 125 370
281,J 174076 125 370
282,J 174100 125 370
283,J 174102 125 370
284,J 174104 311 370
285,J
286,J 174106 235 371
287,J 174110 242 371
288,J
289,J
290,J
291,J

*
CMDTS

*
* SHIFTED COMMAND ROUTINE ADDRESS TABLE
*

DA	REGA	A - REGISTER DISPLAY
DA	REGB	B - REGISTER DISPLAY
DA	REGC	C - REGISTER DISPLAY
DA	REGD	D - REGISTER DISPLAY
DA	REGE	E - REGISTER DISPLAY
DA	REGF	F - CONDITION FLAGS DISPLAY
DA	GETCME	G - * NOT USED *
DA	REGH	H - REGISTER DISPLAY
DA	EXECUI	I - 'E' COMMAND WITH EI/RET
DA	TSTDSP	J - DISPLAY TEST
DA	GETCME	K - * NOT USED * (KEYBOARD TEST)
DA	REGL	L - REGISTER DISPLAY
DA	GETCME	M - * NOT USED *
DA	GETCME	N - * NOT USED *
IFS	ORIGINI	
DA	ORIGINI	O - MODIFY SELECTED ORIGIN
XIF		
IFC	ORIGINI	
DA	GETCME	Q - * NOT USED * (ORIGIN MODIFY)
XIF		
IFNE	ROMTYPE,3	
DA	BASSET	P - DISPLAY BASE REG OR LOAD WITH C
XIF		
IFEQ	ROMTYPE,3	
DA	GETCME	P - * NOT USED * (BASE REGISTER)
XIF		
DA	GETCME	Q - * NOT USED *
IFS	STACKP	
DA	STACKR	R - STACK ROLL
DA	STACKS	S - STACK STUFF
XIF		
IFC	STACKP	
DA	GETCME	R - * NOT USED * (STACK ROLL)
DA	GETCME	S - * NOT USED * (STACK STUFF)
XIF		
DA	GETCME	T - * NOT USED * (PSEUDO RANDOM MEM TEST)
DA	GETCME	U - * NOT USED *
DA	GETCME	V - * NOT USED *
DA	GETCME	W - * NOT USED *
DA	REGX	X - REGISTER DISPLAY
IFNE	ROMTYPE,3	
DA	OUTXST	Y - EX STATUS
DA	OUTXDA	Z - EX DATA
XIF		
IFEQ	ROMTYPE,3	
DA	GETCME	Y - * NOT USED * (5500 I/O BUS)
DA	GETCME	Z - * NOT USED * (5500 I/O BUS)

292 J
293 J
294 J 174112
295 J
296 J
297 J
298 J 174112 062 074 345
299 J 174115 110 124 370
300 J 174120 111 074 024
301 J 174123 053
302 J 174124 060
303 J
304 J 174125
305 J
306 J
307 J
308 J 174125 151
309 J 174126 104 216 367
310 J
311 J 174131
312 J
313 J
314 J
315 J 174131 113 074 106
316 J 174134 150 207 367
317 J 174137 113 024 001
318 J 174142 106 222 372
319 J 174145 104 216 367
320 J
321 J 174150
322 J
323 J
324 J
325 J
326 J 174150 024 060
327 J 174152 160 125 370
328 J 174155 074 010
329 J 174157 120 125 370
330 J 174162 062 202
331 J 174164 111 211
332 J 174166 062 202
333 J 174170 111 211
334 J 174172 062 202
335 J 174174 111 211
336 J 174176 062 260
337 J 174200 113 074 117
338 J 174203 113 014 000
339 J 174206 104 216 367

XIF
*
C12345
*
CHECK FOR '12345' FOR COMMAND ENTRY VALUE
*
CPC 012345 CHECK LSB
JFZ C1234P ABORT IF NOT RIGHT
CPB 012345>8 ELSE CHECK MSB
RTZ RETURN IF BOTH CORRECT
C1234P POP ELSE ABORT THE RETURN
*
GETCME
*
COMMAND ERROR
*
EX BEEP MAKE NOISE IF BAD COMMAND
JMP GETCML
*
BACKSP
*
BACKSPACE IF NOT AT BEGINNING OF LINE
*
CPD NECP DON'T DO ANYTHING IF AT BEGINNING OF LINE
JTZ GETCMD
SUD 1 ELSE DECREMENT THE HORIZONTAL POSITION
CALL SBCRL3 AND SHIFT BC BACK RIGHT THREE PLACES
JMP GETCML THEN CONTINUE THE COMMAND
*
GETDIG
*
CONVERT OCTAL DIGITS ENTERED BEFORE COMMAND LETTER TO BINARY
*
LLA SAVE THE CHARACTER ENTERED
SU 101 SEE IF LEGAL ASCII DIGIT
JTS GETCME COMPLAIN IF NOT
CP 8
JFS GETCME
ADCC ELSE CONVERT TO BINARY
ACBB BY SHIFTING BC LEFT 3
ADCC
ACBB
ADCC
ACBB
ORAC AND OR'ING IN THE DIGIT
CPD MAXPOS=1
ACD 0
JMP GETCML

342,J		IFS	ASCII	
343,J	*			
344,J	KEYINM			
345,J	.			
346,J	. SET ASCII KEYIN MODE			
347,J	.			
348,J		CALL	C12345	ENTRY MUST BE '12345K'
349,J		HL	KEYMSG	DISPLAY KEYIN MODE MESSAGE
350,J		DE	61<8+BL	POSITION CURSOR FOR MESSAGE
351,J		CALL	DISPLAY	DISPLAY OR CLEAR THE MESSAGE
352,J		LA	0377	SET THE KEYIN MODE FLAG
353,J		POP		(KEEP THE STACK CORRECT)
354,J		JMP	KEYINS	
355,J	.			
356,J	KEYINC	HL	KEYMSG	RESET THE KEYIN MODE
357,J		DE	61<8+BL	
358,J		CALL	DISPLAY	BLANK THE LINE
359,J		XRA		
360,J		POP		(KEEP THE STACK CORRECT)
361,J		JMP	KEYINS	DONE
362,J	.			
363,J	KEYMSG	DC	' * ASCII'	KEYIN MODE MESSAGE
364,J	KEYMSC	DC	SEEDL,SES	
365,J	*			
366,J	KEYINA			
367,J	.			
368,J	. PROCESS NEXT CHARACTER IN ASCII KEYIN MODE			
369,J	.			
370,J		CP	CAN	EXIT THE MODE IF CANCEL ENTERED
371,J		JTZ	KEYINC	
372,J		CP	BSP	BACK UP ONE LOCATION IF BACKSPACE ENTERED
373,J		JTZ	DECA DR	
374,J		CP	DEL	GO FORWARD ONE LOCATION IF DEL ENTERED
375,J		JTZ	INCA DR	
376,J		CP	'A'	ELSE DO SHIFT KEY INVERSION
377,J		JTC	KEYINN	
378,J		CP	'Z'+1	
379,J		JTC	KEYINV	
380,J		CP	' '	
381,J		JTC	KEYINN	
382,J		CP	' '+1	
383,J		JFC	KEYINN	
384,J	KEYINV	XR	040	INVERT SHIFT IF ALPHA CHARACTER
385,J	KEYINN	DPL	HL,CURADR	STORE THE ASCII CHAR IN CURRENT ADDRESS
386,J		LMA		AND THEN BUMP THE CURRENT ADDRESS
387,J		XRA		
388,J		EX	CLICK	MAKE SOME NOISE IF ASCII CHARACTER
389,J		XIF		

```

390,J
391,J 174211
392,J
393,J
394,J
395,J 174211 115 164 200
396,J 174214 110 223 370
397,J 174217 026 001 016 000
398,J 174223 176 202
399,J 174225 115 211
400,J 174227 115 166 200
401,J 174232 007
402,J
403,J 174233
404,J
405,J
406,J
407,J 174233 110 241 370
408,J 174236 111 124 200
409,J 174241 174 047
410,J 174243 113 146 200
411,J 174246 007
412,J
413,J 174247
414,J
415,J
416,J
417,J 174247 113 074 106
418,J 174252 053
419,J
420,J
421,J
422,J
423,J
424,J 174253 111 126 200
425,J 174256 007
426,J
427,J 174257
428,J
429,J
430,J
431,J 174257 115 164 200
432,J 174262 110 271 370
433,J 174265 026 001 016 000
434,J 174271 176 222
435,J 174273 115 231
436,J 174275 115 166 200
437,J 174300 007

```

```

*
INCADR
. INCREMENT THE CURRENT ADDRESS BY ONE OR THE VALUE GIVEN
.
      DPL      HL,CURADR
      JFZ      INCADV
      BC       1
INCADV  ADCL
      ACBH
MODIFNXT DPS      HL,CURADR
      RET

*
LINK
. LINK TO THE ADDRESS POINTED TO BY THE CURRENT ADDRESS
.
      JFZ      LINKADR      USE ADDRESS GIVEN
      DPL      BC,CURADR    NO, USE CURRENT ADDRESS
LINKADR DL      DE,BC
      DPS      DE,CURADR
      RET

*
NEWADR
. SET THE CURRENT ADDRESS
.
      CPD      NECP          IGNORE ENTER ONLY
      RTZ
      IFS      ORIGINI
      DPL      DE,CURORG     ELSE ADD ON THE CURRENT ORIGIN
      ADEC      VALUE
      ACDB
      XIF
NOSADR  DPS      BC,CURADR
      RET

*
DECADR
. DECREMENT THE CURRENT ADDRESS BY ONE OR THE VALUE GIVEN
.
      DPL      HL,CURADR
      JFZ      DECADV
      BC       1
DECADV  SUCL
      SBBH
      DPS      HL,CURADR
      RET

```


438,J
439,J
440,J
441,J 174301 035
442,J 174302 035
443,J 174303 035
444,J 174304 035
445,J 174305 035
446,J 174306 035
447,J 174307 035
448,J 174310 035
449,J 174311 035
450,J
451,J 174312 115 166 200
452,J 174315 113 074 106
453,J 174320 053
454,J 174321 104 343 370
455,J
456,J 174324
457,J
458,J
459,J
460,J 174324 113 074 106
461,J 174327 150 335 370
462,J 174332 111 126 204
463,J 174335 111 124 204
464,J
465,J 174340
466,J
467,J
468,J
469,J 174340 051 227 370
470,J
471,J 174343
472,J
473,J
474,J
475,J 174343 115 164 200
476,J 174346 372
477,J 174347 015
478,J 174350 111 261
479,J 174352 053
480,J 174353 371
481,J 174354 015
482,J 174355 153
483,J 174356 007
484,J
485,J
486,J
487,J
488,J
489,J

*
* REGISTER DISPLAYS
*
REGF DECP HL
REGC DECP HL
REGB DECP HL
REGE DECP HL
REGD DECP HL
REGL DECP HL
REGH DECP HL
REGA DECP HL
REGX DECP HL
*
REGDSP DPS HL,CURADR DISPLAY THAT REGISTER CONTENT
CPD NECP JUST DISPLAY IF NO DIGITS ENTERED
RTZ
JMP MODIFY OTHERWISE MODIFY
*
MODAGN
*
* 'A' MODIFY AND INCREMENT THE ADDRESS USING NEW OR PREVIOUSLY SET VALUE
*
CPD NECP CHECK IF DIGITS ARE GIVEN
JTZ MODIFX
DPS BC,MODVAL UPDATE TO NEW VALUE GIVEN
MODIFX DPL BC,MODVAL OR USE OLD VALUES
*
MODINC
*
* 'I' MODIFY AND INCREMENT THE ADDRESS
*
PUSH MODIFNXT RETURN TO SAVE ADDRESS ROUTINE
*
MODIFY
*
* MODIFY THE CURRENTLY ADDRESSED MEMORY LOCATION
*
DPL HL,CURADR STORE LSB WHERE CURRENT ADDRESS POINTS
LMC HL
INCP HL (INCREMENT ADDRESS BY ONE - IF NEEDED)
ORBR
RTZ THAT'S IT IF MSB IS ZERO
LMB ELSE STORE MSB IN FOLLOWING LOCATION
INCP HL (INCREMENT ADDRESS BY TWO - IF NEEDED)
EX CLICK MAKE NOISE IF TWO BYTES STORED
RET
*
IFS ORIGINI
+
ORIGIN
*
* SET THE CURRENT ORIGIN TO THE TABLE ENTRY SELECTED

```

490,J
491,J
492,J
493,J
494,J
495,J
496,J
497,J
498,J
499,J
500,J
501,J
502,J
503,J
504,J
505,J
506,J
507,J
508,J
509,J
510,J
511,J
512,J
513,J
514,J
515,J
516,J
517,J
518,J
519,J
520,J
521,J
522,J
523,J
524,J
525,J
526,J
527,J
528,J
529,J
530,J
531,J
532,J
533,J
534,J
535,J
536,J
537,J
538,J
539,J
540,J
541,J

```

	•			
		JTZ	ORIGIC	JUST CLEAR ORIGIN IF NO SELECT GIVEN
		LAC		ELSE MAKE SURE SELECT IS WITHIN RANGE
		CP	OTABLN	
		JFC	GETCME	
		PS	A,CUROS	SAVE THE ORIGIN NUMBER
ORIGI0	ADA			INDEX INTO THE ORIGIN TABLE
	HL	OTABL		
	INCP	HL,A		
	DL	BC,HL		GET THE NEW ORIGIN VALUE
ORIGIC	DPS	BC,CURORG		SET THE CURRENT ORIGIN TO THAT VALUE
	BC	CURORG		DISPLAY THE SELECTED ORIGIN
	DPS	BC,CURADR		
	RET			
	•			
	ORIGIM			
	•			
	•			MODIFY THE CURRENTLY SELECTED ORIGIN TABLE ENTRY
	•			
		PL	A,CUROS	GET THE CURRENT ORIGIN SELECT NUMBER
		JTZ	ORIGI0	SELECT PREVIOUS ENTRY IF NO DIGITS
		CP	OTABLN	MAKE SURE THE SELECT NUMBER IS WITHIN
		JFC	GETCME	RANGE IF WE'RE GOING TO UPDATE
		ADA		ELSE INDEX INTO ORIGIN TABLE BY
		HL	OTABL	THE CURRENT SELECT NUMBER
		INCP	HL,A	
		DS	BC,HL	STORE THE NEW VALUE THERE
		JMP	ORIGIC	AND SAVE AND DISPLAY IT
		XIF		
	•			
		IFS	STACKP	
	•			
	STACKR			
	•			
	•			ROLL (POP) THE STACK THE NUMBER OF TIMES GIVEN IN THE C-REGISTER
	•			
		LAC		
		JFZ	STACKOV	
		LA	1	
STACKOV	CP	32-1		
	JFC	GETCME		CAN'T POP MORE THAN STACKL/2-1 ENTRIES
	DE	DEBUG		
	DPL	HL,OLDTOS		
	•			
STACKOL	DS	DE,HL		
	NOL	-1-64		
	INCP	HL,2		STACK POPS "UP"
	ORL	64		
	SU	1		
	JFZ	STACKOL		
	JMP	STACKD0		
	•			

```

542,J
543,J
544,J
545,J
546,J
547,J
548,J
549,J
550,J
551,J
552,J
553,J
554,J
555,J 174357
556,J
557,J
558,J
559,J 174357 115 164 207
560,J 174362 302
561,J 174363 074 037
562,J 174365 100 125 370
563,J 174370 200
564,J 174371 176 044 277
565,J 174374 017
566,J 174375 176 064 100
567,J 174400 115 166 200
568,J 174403 007
569,J
570,J 174404
571,J
572,J
573,J
574,J 174404 110 012 371
575,J 174407 111 124 200
576,J 174412 062 307
577,J 174414 074 052
578,J 174416 150 062 371
579,J 174421 066 211 056 327
580,J 174425 006 003
581,J 174427 017
582,J 174430 176 074 251
583,J 174433 100 062 371
584,J 174436 307
585,J 174437 074 377
586,J 174441 110 025 371
587,J 174444 035
588,J 174445 111 027
589,J 174447 117 015
590,J 174451 062 307
591,J 174453 370
592,J 174454 006 052
593,J 174456 062 370

```

STACKS

* STUFF (PUSH) A VALUE ONTO THE STACK

```

*
      DPL      HL,OLDTOS
      DECP     HL,2      MAKE B.O.S. NEW T.O.S.
      ORL      64        STACK GROWS "DOWN"
      DS       BC,HL
*
STACKD0 LC      0        CAUSE TOP STACK ENTRY TO BE DISPLAYED
      OPS     HL,OLDTOS
      XIF

```

* STACKD

* DISPLAY A STACK LOCATION

```

*
      DPL      HL,OLDTOS      (HL)=OLD TOP OF STACK VALUE
      LAC      (C) IS STACK NUMBER
      CP       32-1
      JFC      GETCME
      ADA      DOUBLE FOR INDEX
      NDJ      -1-64
      INCP     HL,A
      ORL      64
      OPS     HL,CURADR
      RETURN

```

* BPSET

* SET A BREAK POINT AT THE CURRENT OR GIVEN ADDRESS

```

*
      JFZ      BPSETG      USE CURRENT ADDRESS IF NO DIGITS GIVEN
      DPL      BC,CURADR
BPSETG LAM      BC        DON'T ALLOW BP IF BP ALREADY THERE
      CP       OPCODEBP
      JTZ      BPTSER
      HL       BPTABL=BPTES+1 SEARCH THE TABLE FOR AN OPENING
BPSRCH LA       BPTES     BUMP TO THE NEXT TABLE OPENING
      INCP     HL,A
      CPL      BPTABE
      JFC      BPTSER     STOP IF PAST END OF TABLE
      LAM      ERROR IF NO SLOTS AVAILABLE
      CP       GET THE MSB OF THE ADDRESS
      -1       KEEP GOING IF NOT A FREE SLOT
      JFZ      BPSRCH
      DECP     HL        ELSE STORE THE ADDRESS IN THE TABLE
      DS       BC,HL
      INCP     HL,2      THEN SAVE WHAT WAS THERE IN MEMORY
      LAM      BC
      LMA      OPCODEBP THEN STORE THE BREAK POINT INSTRUCTION
      LMA      BC

```

Address	Op Code	Op Name	Comments
594,J	174460	153	EX CLICK
595,J	174461	007	RET MAKE A LITTLE NOISE
596,J			DON'T CHANGE THE CURRENT ADDRESS
597,J	174462	151	BPTSER EX BEEP
598,J	174463	104 253 370	JMP TELL THAT CAN NOT BREAKPOINT THERE
599,J			SHOW BP ALREADY THERE (OR NO TABLE ROOM)
600,J	174466		BPCLR
601,J			.
602,J			. CLEAR ALL THE BREAK POINTS
603,J			.
604,J	174466	066 213 056 327	HL BPTABL
605,J	174472	176 074 251	CPL BPTABE
606,J	174475	150 060 373	JTZ BPTCLR
607,J	174500	047	DL DE,HL
608,J	174501	117 015	INCP HL,2
609,J	174503	307	LAM
610,J	174504	015	INCP HL
611,J	174505	113 074 377	CPD =1
612,J	174510	150 072 371	JTZ BPCRL
613,J	174513	174 370	LMA DE
614,J	174515	104 072 371	JMP BPCRL
615,J			ELSE RESTORE THE CONTENTS
616,J	174520		AND MOVE ON TO THE NEXT ENTRY
617,J			REGMOD
618,J			.
619,J			. SWITCH REGISTER MODES (ALPHA/BETA)
620,J	174520	176 064 077	ORL SEPSWSV AND 0177 (VALUE 077 OR AN ERROR)
621,J	174523	115 166 200	DPS HL,CURADR
622,J	174526	006 200	LA SWALBT
623,J	174530	104 311 372	JMP SREGMOD
624,J			GO TO SET THE REGISTER MODE
625,J			IFNE ROMTYPE,3
626,J			BASSET
627,J	174533		.
628,J			. SET OR DISPLAY THE BASE REGISTER
629,J			.
630,J			HL SEBRLS
631,J	174533	066 253 056 357	JTZ BASED
632,J	174537	150 154 371	CP 'P'
633,J	174542	074 120	LAC
634,J	174544	302	JTZ BASSES
635,J	174545	150 153 371	.
636,J			LAB
637,J	174550	301	SU
638,J	174551	024 200	0100000>8
639,J			.
640,J	174553	072	BASSES BRL
641,J	174554	115 166 200	BASED DPS
642,J	174557	007	RET HL,CURADR
			THEN STORE AND UPDATE THE BASE REGISTER
			CAUSE THE BASE REGISTER STORAGE
			TO BE DISPLAYED

643,J
644,J 174560
645,J
646,J
647,J
648,J
649,J 174560 106 112 370
650,J 174563 115 164 200
651,J 174566 307
652,J 174567 074 020
653,J 174571 100 125 370
654,J 174574 320
655,J 174575 015
656,J 174576 077
657,J 174577 007

*
STLOAD
*
* LOAD THE SECTOR TABLE - LOAD (N) ENTRIES WHERE (N) IS THE BYTE POINTED TO
* BY THE CURRENT ADDRESS AND THE ENTRIES FOLLOW THAT LOCATION.
*
CALL C12345 ENTRY MUST BE '123450'
DPL HL,CURADR IF '123450' THEN LOAD FROM CURADR
LAM A = NUMBER OF ENTRIES
CP 16 ERROR IF NUMBER OF ENTRIES >15
JFC GETCME
LCA
INCP HL SET C TO NUMBER OF ENTRIES
STL RUMP HL TO START OF LIST
RET I SURE HOPE EVERYTHING WAS SET UP RIGHT!

658,J
659,J 174600
660,J
661,J
662,J
663,J
664,J
665,J
666,J
667,J 174600 110 210 371
668,J 174603 066 252 056 357 327
669,J 174610
670,J 174610 062 121
671,J 174612 074 141
672,J 174614 150 223 371
673,J
674,J 174617 125
675,J 174620 074 147
676,J 174622 053
677,J
678,J 174623 101
679,J 174624 066 203 056 327 370
680,J 174631 115 166 200
681,J 174634 007
682,J
683,J
684,J
685,J
686,J
687,J 174635 106 300 371
688,J 174640 123
689,J 174641 007
690,J
691,J 174642 106 300 371
692,J 174645 125
693,J 174646 007
694,J
695,J 174647 106 300 371
696,J 174652 127
697,J 174653 007

*
ADRDEV
.
* ADDRESS AN I/O DEVICE AND DISPLAY ITS CURRENT STATUS
.
* NOTE: MINOR DIFFERENCE FROM 5500 VERSION FOR 'F' AND 'G' MODES
* THE FIRST INPUT BY THIS CODE GIVES ACTUAL DATA NOT STATUS.
* DIFFERENCE FOUND BY GADWA'S GROUP
.
JFZ ADRDE1 ADDRESS IS GIVEN
MLC *SEXADR USE LAST ADDRESSED DEV IF NOT GIVEN
ADRDE1 ADDRESS THE DEVICE
EXC ADR (AUTOMATICALLY SETS LAST ADDRESS GIVEN)
CP ' ' DO INPUT IF 'A' COMMAND
JTZ GETSTA
.
EX DATA ELSE PUT DEVICE IN DATA MODE
CP ' ' JUST EXIT IF 'G' COMMAND
RTZ
*
GETSTA IN GET THE STATUS OF THAT DEVICE
MSA *CURSTA
DPS HL,CURADR CAUSE THE STATUS TO BE DISPLAYED
RET
*
* ADDRESS THE CURRENTLY ADDRESSED I/O DEVICE AND OUTPUT THE VALUE GIVEN
* OR THE LAST VALUE GIVEN FOR OUTPUT DOING AN I/O STROBE BASED
* UPON THE COMMAND LETTER GIVEN.
*
OUTXST CALL OUTSETUP ' ' - EX STATUS
EX STATUS
RET OFF TO "GETSTA"
*
OUTXDA CALL OUTSETUP ' ' - EX DATA
EX DATA
RET OFF TO "GETSTA"
*
OUTXWT CALL OUTSETUP 'W' - EX WRITE
EX WRITE
RET OFF TO "GETSTA"

698,J
699,J 174654 106 300 371
700,J 174657 131
701,J 174660 007
702,J
703,J 174661 106 300 371
704,J 174664 133
705,J 174665 007
706,J
707,J 174666 106 300 371
708,J 174671 135
709,J 174672 007
710,J
711,J 174673 106 300 371
712,J 174676 137
713,J 174677 007
714,J
715,J 174700
716,J
717,J
718,J
719,J
720,J 174700 060
721,J 174701 051 223 371
722,J 174704 070
723,J 174705 105 202
724,J 174707 053
725,J
726,J 174710 302
727,J 174711 126 202
728,J 174713 007
729,J

*
OUTXC1 CALL OUTSETUP 'X' - EX COM1
EX COM1
RET OFF TO "GETSTA"
*
OUTXC2 CALL OUTSETUP 'Y' - EX COM2
EX COM2
RET OFF TO "GETSTA"
*
OUTXC3 CALL OUTSETUP 'Z' - EX COM3
EX COM3
RET OFF TO "GETSTA"
*
OUTXC4 CALL OUTSETUP 'V' - EX COM4
EX COM4
RET OFF TO "GETSTA"
*
OUTSETUP
*
PUSH "GETSTA" ON THE STACK BELOW THE RETURN ADDRESS AND PUT THE RIGHT
VALUE IN THE A-REGISTER FOR THE OUTPUT INSTRUCTION.
*
POP HL
PUSH GETSTA
PUSH HL
PL A,CUROUT
RTZ
*
LAC
PS C,CUROUT
RET
XIF

730 J
731 J 174714
732 J
733 J
734 J
735 J 174714 006 005
736 J 174716 104 330 371
737 J
738 J 174721
739 J
740 J
741 J
742 J 174721 006 001
743 J 174723 104 330 371
744 J
745 J 174726
746 J
747 J
748 J
749 J 174726 006 000
750 J
751 J 174730 115 164 207
752 J 174733 150 340 371
753 J
754 J 174736 111 027
755 J 174740 176 044 300
756 J 174743 035
757 J 174744 267
758 J 174745 370
759 J 174746 106 344 372
760 J
761 J 174751 115 164 207
762 J 174754 176 065
763 J 174756 062 030
764 J
765 J 174760
766 J
767 J
768 J
769 J
770 J 174760 176 064 077
771 J 174763 347
772 J 174764 174 044 373
773 J 174767 374
774 J 174770 113 074 106
775 J
776 J 174773
777 J
778 J
779 J
780 J
781 J 174773 150 001 372

*
EXEUSR
*
* CONTINUE EXECUTION FORCING USER MODE AND INTERRUPTS ENABLED
*
* LA SWUSER+SWINTE USER MODE,INT, ENABLED
* JUMP EXECEM
*
EXECUI
*
* CONTINUE EXECUTION FORCING INTERRUPTS ENABLED
*
* LA SWINTE INTERRUPTS ENABLED
* JUMP EXECEM
*
EXECUT
*
* CONTINUE EXECUTION WITH NO CHANGE IN MODE OR INTERRUPT FLAGS
*
* LA 0 NO CHANGE (CAN'T BE XRA OR JTZ FAILS)
*
EXECEM DPL HL,OLDTOS
JTZ EXECDM CATCH NO COMMAND VALUE GIVEN
*
* DS BC,HL ELSE SET NEW RETURN ADDRESS
EXECDM NDL 0300
DECP HL POINT TO PSW
ORM SET MODE AS DESIRED
LMA HL
CALL RESTRLOC CORRECT REGISTER LOCATIONS
*
* DPL HL,OLDTOS RESTORE USER SYSTEM SAVE AREA
SYSMOV HL
SYSRET LOAD STATE, REGS, AND RETURN
*
CALL
*
* WARNING: 'CALL' MUST NOT BE IN USER MODE OR THE RETURN WILL GENERATE
* AN ERROR BECAUSE MACRO-ROM IS ACCESS PROTECTED
*
* ORL SEPSWSV.AND.0177 POINT TO THE SAVED PSW
* LEM (MAY NOT USE A-REG)
* NDE -1-SWUSER AND TURN OFF THE USER MODE BIT
* LME
* CPD NECP RESTORE THE CC FOR FOLLOWING CODE
*
JUMP
*
* PERFORM A CALL ("DEBUG" IS PUT ON THE STACK AS THE RETURN ADDRESS) OR A
* JUMP TO A GIVEN MEMORY LOCATION OR THE CURRENTLY ADDRESSED MEMORY LOCATION
*
* JTZ CALLKA

782,J
783,J 174776 111 126 200
784,J 175001 022 070
785,J 175003 106 344 372
786,J 175006 022 060
787,J 175010 115 164 207
788,J 175013 111 124 200
789,J 175016 046 127 036 367
790,J 175022 176 065
791,J 175024 074 152
792,J 175026 150 033 372
793,J 175031 174 070
794,J 175033 062 070
795,J 175035 062 030

	DPS	BC,CURADR	SET ADDRESS TO CALL (JUMP TO)
CALLKA	PUSH	XA	(SAVE A - INPUT DEBUG OPERATION)
	CALL	RESTRLOC	CORRECT REGISTER LOCATIONS
	POP	XA	
	DPL	HL,OLDTOS	
	DPL	BC,CURADR	GET CALL (JUMP) ADDRESS
	DE	DEBUG	GET RETURN ADDRESS (IF CALL)
	SYSMOV	HL	RESTORE USER SYSTEM SAVE AREA
	CP	' '	JUMP OPERATION?
	JTZ	JUMPIT	
JUMPIT	PUSH	DE	NO, 'CALL' SO PUSH RETURN ADDRESS
	PUSH	BC	
	SYSRET		LOAD STATE, REGS, AND RETURN

796 J
797 J 175037
798 J
799 J
800 J
801 J 175037 066 117
802 J 175041 150 045 372
803 J 175044 362
804 J 175045 070
805 J 175046 106 004 374
806 J 175051 060
807 J 175052 316
808 J 175053 076 357
809 J 175055 106 053 374
810 J
811 J 175060 106 125 373
812 J 175063 150 060 372
813 J 175066 104 004 374

+
TSTDSP

• TEST THE DISPLAY SCREEN BY FILLING IT FULL OF CHARACTERS

	LL	'0'	STANDARD CHARACTER
	JTZ	TDSTD	
	LLC		WILL NOT BE USED, VALUE ENTERED WILL BE
TDSTD	PUSH	HL	SAVE IT
	CALL	DSPINIT	INIT THE DISPLAY AND ITS POINTERS
	POP	HL	
	LRL		PUT IN B THE CODE TO BE DISPLAYED
	LX	SEDOPTS>8	
	CALL	DOSF68LP	CHANGE THE DISPLAY TO ALL 'B'
TDWAIT	CALL	DOSF61	WAIT FOR ANY KEYIN
	JTZ	TDWAIT	
	JMP	DSPINIT	RE-INIT THE DISPLAY AND RETURN TO GET KEY

814 J
815 J 175071
816 J
817 J
818 J
819 J 175071 066 130 056 372
820 J 175075 106 341 374
821 J
822 J 175100 111 010
823 J 175102 321
824 J 175103 036 017
825 J 175105 106 115 372
826 J
827 J 175110 111 010
828 J 175112 320
829 J 175113 036 013
830 J 175115
831 J 175115 016 000
832 J 175117 066 003
833 J 175121 106 160 372
834 J 175124 006 072
835 J 175126 370
836 J 175127 007
837 J
838 J 175130
839 J
840 J
841 J
842 J 175130 224
843 J 175131 040 115 072
844 J 175134 060
845 J 175135 061
846 J 175136 061
847 J 175137 040
848 J 175140 120
849 J 175141 040 040 040 040
850 J 000013
851 J 175145 040 040 040 040
852 J 000017
853 J 175151 040 040 003
854 J 000012

*
IDENT
*
* DISPLAY THE MICRO AND MACRO ROM VERSION INFORMATION
*
* HL IDENTM DISPLAY THE MACRO ROM VERSION INFORMATION
* CALL DISPLAY HOPING B-REG IS ZERO
*
* INFO C = MICRO-ROM VERSION
* LCB
* LD IDMIC DISPLAY IT IN OCTAL
* CALL IDENT0 (SAVING A LITTLE SPACE)
*
* INFO C = PROCESSOR TYPE
* LCA
* LD IDPROC DISPLAY IT IN OCTAL
IDENT0
LB 0
LL 3
CALL DSPOCT
LA ' '
LMA
RET
*
IDENTM
*
* ROM VERSION IDENTIFICATION MESSAGE
*
IDENTS DC \$HD
DC ' M: '
DC \$MACROM/64.AND.7+'0 '
DC \$MACROM/8.AND.7+'0 '
DC \$MACROM.AND.7+'0 '
DC \$MACVER PUT REVISION LETTERS IN MESSAGE
DC 'P '
DSPBLNK DC ' ' (INVISIBLE COLON APPEARS WHEN NEEDED)
IDPROC EQU \$-IDENTS-1
DC ' '
IDMIC EQU \$-IDENTS-1
DC ' ',SES
DSPBLN EQU \$-DSPBLNK-1

857 J
858 J
859 J
860 J
861 J
862 J
863 J
864 J
865 J
866 J
867 J
868 J
869 J
870 J
871 J 175154 066 006
872 J 175156 036 117
873 J
874 J 175160 306
875 J 175161 022 070
876 J 175163 106 312 374
877 J 175166 022 060
878 J
879 J 175170 022 070
880 J 175172 302
881 J 175173 044 007
882 J 175175 004 060
883 J 175177 370
884 J 175200 035
885 J 175201 106 222 372
886 J 175204 113 024 001
887 J 175207 022 060
888 J 175211 024 001
889 J 175213 110 170 372
890 J
891 J 175216 016 040
892 J 175220 371
893 J 175221 007
894 J
895 J 175222
896 J
897 J
898 J
899 J 175222 111 032
900 J 175224 062 032
901 J 175226 111 032
902 J 175230 062 032
903 J 175232 111 032
904 J 175234 062 032
905 J 175236 111 044 037
906 J 175241 007

*
* DISPLAY OCTAL VALUE RIGHT TO LEFT
*
* ENTRY: BC = VALUE TO BE DISPLAYED
* E = LINE FOR THE DISPLAY (-12 THRU 11)
* L = NUMBER OF CHARACTERS (IF ENTERED BELOW DSPOC6)
* D = POSITION OF RIGHT-HAND CHARACTER
* (IF ENTERED BELOW DSPOCR)
*
* EXITS: BC = SCRATCHED
* D = POSITION TO LEFT OF MOST SIG. DIGIT (BLANKED)
* E = ENTRY VALUE
* HL = SCRATCHED
*
DSPOC6 LL 6 DISPLAY 6 DIGITS
DSPOCR LD 79 DISPLAY AT RIGHT OF SCREEN
*
DSPOCT LAL GET THE COUNT IN THE A-REGISTER
PUSH XA
CALL DSPCBL POSITION HL INTO DISPLAY BUFFER
POP XA
*
DSPOCL PUSH XA SAVE THE COUNT ON THE STACK
LAC GET THE LEAST SIGNIFICANT DIGIT
ND 7 ISOLATE THE OCTAL BITS
AD '0' CONVERT TO ASCII DIGIT
LMA DISPLAY THE ASCII DIGIT
DECP HL
CALL SBCRL3 SHIFT THE 16-BIT VALUE RIGHT 3 BITS
SUD 1 DECREMENT THE HORIZONTAL POSITION
POP XA DECREMENT THE COUNT
SU 1
JFZ DSPOCL LOOP IF MORE DIGITS TO GO
*
LB ' ' THEN BLANK THE POSITION TO THE LEFT
LMB
RET
*
SBCRL3
*
* SHIFT BC RIGHT LOGICALLY 3 BITS
*
SREB
SREC
SREB
SREC
SREB
SREC
NDB 037
RET

907,J
908,J 175242
909,J
910,J
911,J
912,J
913,J 175242 062 020
914,J 175244 066 200 056 327
915,J 175250 174 060
916,J 175252 176 065
917,J 175254 174 070
918,J 175256 076 327
919,J 175260 117 275
920,J 175262 053
921,J
922,J 175263 115 166 207
923,J 175266 176 044 300
924,J 175271 117 035
925,J 175273 106 363 372
926,J 175276 035
927,J 175277 106 325 372
928,J 175302 035
929,J 175303 106 325 372
930,J
931,J 175306 176 064 077
932,J 175311 257
933,J 175312 370
934,J 175313 120 321 372
935,J
936,J 175316 006 011
937,J 175320 037
938,J 175321 115 166 211
939,J 175324 007
940,J
941,J 175325
942,J
943,J
944,J
945,J 175325 006 004
946,J
947,J 175327 047
948,J 175330 324
949,J 175331 343
950,J 175332 332
951,J 175333 027
952,J 175334 117 035
953,J 175336 024 001
954,J 175340 110 327 372
955,J
956,J 175343 007

+
SSTATE
.
.
.
.
.
SYSSAV
HL SDBGWS
POP DE
SYSMOV HL
PUSH DE
LX SDBGWS>8
CPHX
RTZ
.
DPS HL,OLDTOS
NDL 0300
DECP HL,2
CALL SWITZAF
DECP HL
CALL RSWAP
DECP HL
CALL RSWAP
.
SREGMOD ORL SEPSWSV,AND,0177
XRM HL
LMA HL
JFS SSTAT1
.
LA 9
DECP HL,A
SSTAT1 DPS HL,OLDREGS
RET
*
RSWAP
.
.
.
SWAP THE LOCATIONS OF LSB AND MSB REGISTERS IN PAIRS IN MEMORY
.
LA 4
.
RSWL DL DE,HL
LCE
LED
LDC
DS DE,HL
DECP HL,2
SU 1
JFZ RSWL
.
RETURN

SAVE CURRENT MODE OF REGISTERS
POINT TO NEW STACK AREA
(RETURN ADDRESS)
MOVE SYSTEM SAVE AREA
(TO NEW STACK)

IS STACK THE DEBUG STACK ?
YES, DONE.

POINT TO ALPHA (F)

POINT TO ALPHA (AX)

POINT TO BETA AX
(RETURNS A=0)

BACK UP TO PSW
GET PSW (SETTING NEW MODE IF REGMOD)

WANT BETA REGS

957,J
958,J 175344
959,J
960,J
961,J
962,J
963,J
964,J
965,J 175344 176 064 077
966,J 175347 117 035
967,J 175351 106 325 372
968,J 175354 035
969,J 175355 106 325 372
970,J 175360 006 022
971,J 175362 017
972,J
973,J
974,J 175363
975,J
976,J
977,J
978,J
979,J 175363 006 011
980,J 175365 337
981,J 175366 037
982,J 175367 111 047
983,J 175371 037
984,J 175372 015
985,J 175373 347
986,J 175374 372
987,J 175375 035
988,J 175376 017
989,J 175377 027
990,J 175400 017
991,J 175401 371
992,J 175402 007

*
RESTRLOC
.
HL POINTS INTO REG-SAVE AREA (OR PSW) ON ENTRY
.
RESTORE THE LOCATIONS OF THE REGISTERS TO THEIR NORMAL
PLACES IN THE STATE STORAGE AREA IN MEMORY.
.
ORL SEPSWSV.AND.0177 (VALUE SHOULD BE 077 OR AN ERROR)
DECP HL,2 POINT TO ALPHA XA PAIR SWAPPED
CALL RSWAP
DECP HL
CALL RSWAP
LA 18
INCP HL,A
.
FALLS THROUGH INTO SWITZAF
*
SWITZAF
.
SWITCH THE A REGISTER AND FLAGS IN MEMORY FOR BOTH MODES
ON ENTRY, MUST POINT TO ALPHA FLAGS (TO BECOME A-REG AS IT WAS)
.
LA 9
LDM D = ALPHA FLAGS
DECP HL,A
DL BC,HL BC = ALPHA A-REG AND BETA FLAGS
DECP HL,A
INCP HL
LEM E = BETA A-REG
LMC STORE BETA FLAGS THERE
DECP HL
INCP HL,A
DS DE,HL STORE BETA A-REG AND ALPHA FLAGS
INCP HL,A
LMB STORE ALPHA A-REG
RET

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- DEBUG SUBROUTINES -

```

993,J
994,J 175403
995,J
996,J
997,J
998,J
999,J 175403 174 060
1000,J 175405 066 227 056 363
1001,J 175411 026 015
1002,J 175413 077
1003,J 175414 066 225 056 363
1004,J 175420 026 322
1005,J 175422 062 077
1006,J 175424 250
1007,J 175425 072
1008,J 175426 174 070
1009,J
1010,J 175430 066 270 056 360
1011,J 175434 046 000 036 357
1012,J 175440 026 124
1013,J
1014,J 175442 310
1015,J 175443 021
1016,J 175444 007
1017,J
1018,J 175445
1019,J
1020,J
1021,J
1022,J
1023,J
1024,J
1025,J
1026,J
1027,J 175445 106 003 373
1028,J 175450 060
1029,J 175451 046 300 036 357
1030,J 175455 174 065
1031,J 175457 070
1032,J
1033,J
1034,J
1035,J
1036,J
1037,J
1038,J
1039,J
1040,J
1041,J
1042,J
1043,J
1044,J

```

```

*
MTSETUP
.
. INITIALIZE: THE SECTOR TABLE, THE BASE REGISTER TO ZERO, AND
. THE INTERRUPT VECTOR TO THE INTERNAL TRAP MESSAGES.
.
*
MTSETX POP DE ** INSURE CALLER GETS RETURNED TO
HL UMST INIT THE SECTOR TABLE FOR USER
LC UMSTL MEMORY SECTORS
STL
HL SMST INIT THE SECTOR TABLE FOR SYSTEM
LC (SEDSPBF>8,AND,0360)+SMSTL MEMORY SECTORS
STLOC
XRA ZERO THE BASE REGISTER
BRL
PUSH DE ** IF THE STACK GETS MOVED (1800MOVI)
.
HL VECTI INIT THE INTERRUPT VECTOR
DE SVMEMP
LC VECTIL
XRA
LBA
BT
RET
.
*
SETUP
.
. INITIALIZE: THE SECTOR TABLE, THE BASE REGISTER TO ZERO, THE INTERRUPT
. VECTOR TO THE INTERNAL TRAP MESSAGES, THE STACK AREA TO
. THE "NORMAL" SYSTEM RAM AREA, THE ORIGIN TABLE TO
. ALL ZEROS, AND THE BREAK POINT TABLE TO ALL FREE ENTRIES.
. WARNING: IF SECTOR TABLE MOVES, ROUTINE MTSETUP STOPS RETURN TO CALLER.
.
*
CALL MTSETUP
POP HL SAVE THE RETURN ADDRESS IN HL
DE SESTACK MOVE THE STACK AREA TO THE
SYSMOV DE "NORMAL" SYSTEM RAM AREA
PUSH HL RESTORE THE RETURN ADDRESS
*
IFS ORIGINI
ORGCLR
.
. CLEAR ALL ORIGIN TABLE ENTRIES
.
HL CURORG
XRA
LMA HL
INCP HL
CPL OPTABE
JFZ OTCLOP
XIF
OTCLOP

```

1045 J
1046 J 175460
1047 J
1048 J
1049 J
1050 J 175460 066 213 056 327
1051 J 175464 006 377
1052 J 175466 370
1053 J 175467 015
1054 J 175470 176 074 251
1055 J 175473 110 066 373
1056 J 175476 007
177
178

*
BPTCLR
.
FREE ALL BREAK POINT ENTRIES
.
HL BPTABL
LA =1
BPTCLL LMA
INCP HL
CPL BPTABE
JFZ BPTCLL
RET
.
INC 1800DSPR

3,K
4,K
5,K
6,K
7,K
8,K
9,K
10,K
11,K
12,K
13,K
14,K
15,K
16,K
17,K
18,K
19,K
20,K
21,K
22,K
23,K
24,K
25,K
26,K
27,K
28,K
29,K
30,K
31,K
32,K
33,K
34,K
35,K
36,K
37,K
38,K
39,K
40,K
41,K
42,K
43,K
44,K
45,K
46,K
47,K
48,K
49,K
50,K
51,K
52,K
53,K
54,K

1800 DISPLAY INTERFACE ROUTINES JUNE 8, 1978 11:53 HSP/HJS

GENERAL NOTES ON THE FOLLOWING ROUTINES:

THE FLASHING OF THE CURSOR IS CONTROLLED BY THE "CURONOFF" AND "CURINBUF" BITS IN THE "SECFLAGS" BYTE. THE POSITION OF THE CURSOR IS DETERMINED BY THE TWO BYTES IN "SECPOS" (VERTICAL) AND "SECPOS"+1 (HORIZONTAL).

WHEN THE "CURONOFF" BIT IS A ZERO, THE CURSOR IS NOT FLASHED. OTHERWISE, WHEN THE "SELFREQ" BIT IN THE "CURINBUF" BIT POSITION IS DIFFERENT FROM THE "CURINBUF" BIT, THE CURSOR IS EITHER:
1) TURNED ON IF THE "SELFREQ" BIT IS NOW A ONE OR 2) TURNED OFF IF THE "SELFREQ" BIT IS NOW A ZERO. WHEN THE CURSOR IS TURNED ON, THE CHARACTER AT THE CURSOR POSITION IS FIRST SAVED INTO "SECHIDE" AND THE CHARACTER AT "SECCHAR" IS STORED WHERE THE CURSOR POSITION DIRECTS, PUTTING IT "IN THE BUFFER".

WHEN THE CURSOR IS TURNED OFF, IF THE CHARACTER AT "SECCHAR" IS NOT THE SAME AS THAT WHERE THE CURSOR POSITION DIRECTS IT IS ASSUMED THAT THE CURSOR WAS OVERWRITTEN BY SOME ROUTINE AND THE CHARACTER AT "SECHIDE" IS NOT RESTORED AS IS NORMALLY DONE.

WHEN THE "CURINBUF" BIT IS A ONE, THE CHARACTER IN THE CURSOR POSITION IS THE CURSOR CHARACTER. OTHERWISE, IT IS THE CHARACTER THAT IS TO NORMALLY BE DISPLAYED IN THAT POSITION. THE EXCEPTION TO THIS RULE IS WHEN THE CURSOR POSITION IS NOT WITHIN THE CONFINES OF THE SCREEN (-12 TO +11 VERTICALLY AND 0 TO 79 HORIZONTALLY), THE CURSOR IS NOT FLASHED.

BEFORE MANY OF THE FOLLOWING ROUTINES WRITE TO THE SCREEN, THE CURSOR IS SUSPENDED. TO SUSPEND THE CURSOR, NO ACTION IS REQUIRED IF THE "CURINBUF" BIT IS A ZERO BECAUSE THE CURSOR CHARACTER IS NOT ON THE SCREEN. OTHERWISE, THE "CURONOFF" BIT IS RESET TO ZERO AND THE CHARACTER AT "SECHIDE" IS STORED WHERE THE CURSOR POSITION DIRECTS IF THE CURSOR CHARACTER "SECCHAR" WAS THERE.

WARNING: THE FOLLOWING ROUTINES THAT CALL 'BLINK' OR 'CURSES' ARE NOT RE-ENTRANT. I.E. THEY CAN NOT BE USED IN A MULTI-TASKING ENVIRONMENT UNLESS ALL OF THEM ARE IN THE SAME TASK (ALL FOREGROUND OR ALL BACKGROUND). IF THIS IS NOT DONE, THE CHARACTER AT THE CURSOR POSITION CAN GET PERMANENTLY CHANGED TO THE CURSOR CHARACTER. THIS IS NOT DESIRABLE. (DISPLAY IS INCLUDED IN THIS RESTRICTION)

ALSO: THE ROUTINES THAT CALL 'CURSESRS' STORE REGISTERS IN A FIXED STORAGE AREA. IF THESE ROUTINES ARE RE-ENTERED, THE REGISTERS AS STORED FROM THE FIRST CALL WILL BE LOST. THE BASIC DEBUG ROUTINES (IN THE ROM) HAVE BEEN MODIFIED SO THAT THIS WILL NOT NORMALLY HAPPEN DUE TO A DEBUG ENTRY. BUT, NOT ALL ROUTINES COULD BE MODIFIED SO, KNOWLEDGE & CARE MUST BE TAKEN IN USING THE DEBUGGER.

57,K
58,K
59,K
60,K
61,K
62,K
63,K
64,K
65,K
66,K
67,K
68,K
69,K
70,K
71,K
72,K
73,K
74,K
75,K
76,K
77,K
78,K
79,K 175477
80,K
81,K
82,K
83,K
84,K
85,K
86,K
87,K
88,K
89,K
90,K
91,K
92,K 175477 106 244 373
93,K 175502
94,K 175502 066 151 056 357 307
95,K 175507 044 003
96,K 175511 004 121
97,K 175513 360
98,K 175514 056 373
99,K 175516 307
100,K 175517 200
101,K 175520 007

```

*
. DOS FUNCTION 6
.
.   SUBFUNCTION: 0 - RETURN KBD/DSP KEY STATUS
.                 1 - CHECK FOR CHARACTER READY
.                 2 - GET A BYTE FROM THE KEYBOARD
.                 3 - WRITE THE BYTE IN (B) TO THE SCREEN
.                 4 - RETURN HOME-UP POSITION IN (DE)
.                 5 - RETURN HOME-DOWN POSITION IN (DE)
.                 6 - TURN CURSOR ON
.                 7 - ROLL UP SCREEN
.                 8 - ERASE FROM CURSOR TO END OF FRAME
.                 9 - ERASE FROM CURSOR TO END OF LINE
.                10 - ROLL DOWN SCREEN
.                11 - TURN CURSOR OFF
.                12 - RETURN FUNCTION KEY & KEYBOARD STATUS
.   DSPCBL - CONVERT CURSOR TO BUFFER POSITION
.   CURSES - SUSPEND THE CURSOR (REMOVE IT FROM BUFFER)
.   BLINK - BLINK THE CURSOR
.   BLINKDE - MOVE CURSOR TO DE AND BLINK IT THERE
.   62N - DOS FUNC 6 SUB 2 WITHOUT "DE" AND CURSOR BLINK
*
DOSF60
.
.   DOS FUNCTION: 6   SUBFUNCTION: 0
.
.   - RETURN THE STATES OF THE "KEYBOARD" AND "DISPLAY" KEYS -
.
.   ENTRY: N/A
.
.   EXITS: SIGN TRUE IF "KEYBOARD" KEY IS DOWN
.          PARITY TRUE IF "DISPLAY" KEY IS DOWN
.          B & X ONLY REGISTERS SAVED
.          2 EXTRA STACK LEVELS USED (MAX)
.
.   CALL      BLINK      BLINK THE CURSOR WHERE IT WAS
.   DOSF60N      (SPECIAL - MACRO-ROM NON BLINK ENTRY)
.   MLA      *SEKBS1      GET KEYBOARD STATUS 1 INTO (A)
.   ND      SEDSPKY,OR,SEKBDKY
.   AD      CCTAB60      INDEX INTO THE TABLE OF BYTES
.   LLA      WHICH WILL GIVE THE CORRECT
.   LH      CCTAB60>8      CONDITION CODE WHEN ADDED
.   LAM      TO THEMSELVES.
.   ADA      SET THE CONDITION CODE
.   RET

```

```

102,K
103,K
104,K
105,K
106,K
107,K
108,K
109,K
110,K 175521 003
111,K 175522 002
112,K 175523 101
113,K 175524 100
114,K
115,K 175525
116,K
117,K
118,K
119,K
120,K
121,K
122,K
123,K
124,K
125,K
126,K
127,K 175525 106 244 373
128,K 175530 066 151 056 357 307
129,K 175535 044 004
130,K 175537 007
131,K
132,K 175540
133,K
134,K
135,K
136,K
137,K
138,K
139,K
140,K
141,K
142,K
143,K
144,K
145,K
146,K
147,K
148,K
149,K
150,K 175540 106 221 373
151,K 175543
152,K 175543 066 151 056 357 307
153,K 175550 044 004

```

```

*
. THE CONDITION CODE GENERATION TABLE IS PAGE SENSITIVE
.
      IFGT      ($,AND,255),252
      RPT      =$,AND,255
      DC      000      PAD TO END OF PAGE
      XIF
.
CCTAB60  DC      0003      FALSE SIGN, PARITY & NO KEYS DOWN
        DC      0002      TRUE PARITY & DISPLAY KEY DOWN
        DC      0101      TRUE SIGN & KEYBOARD KEY DOWN
        DC      0100      TRUE SIGN, PARITY & BOTH KEYS DOWN
*
DOSF61
. DOS FUNCTION: 6 SUBFUNCTION: 1
. - CHECK THE KEYBOARD FOR CHARACTER READY -
.
      ENTRY: N/A
.
      EXITS: FALSE ZERO IF KEYBOARD HAS A CHARACTER
              B & X ONLY REGISTERS SAVED
              2 EXTRA STACK LEVELS USED (MAX)
.
      CALL      BLINK      BLINK THE CURSOR WHERE IT WAS
      MLA      *SEKBS1     GET KEYBOARD STATUS 1 INTO (A)
      ND      SEKBRDY      SET THE ZERO CONDITION FALSE IF
      RET      THE KEYBOARD HAS A CHARACTER
*
DOSF62
. DOS FUNCTION: 6 SUBFUNCTION: 2
. - GET A TRANSLATED CHARACTER FROM THE KEYBOARD -
.
      ENTRY: D = HORIZONTAL CURSOR POSITION
              E = VERTICAL CURSOR POSITION
              D,E NOT USED IN DOSF62N ($KEYCHAR) ENTRY
.
      EXITS: ZERO TRUE IF NO CHARACTER READY
              A = CHARACTER, AFTER TRANSLATION
              BC = SCRATCHED (NOT ON DOSF62N = $KEYCHAR - ENTRY)
              HL = SCRATCHED
      NOTE: L = LOW 7 BITS OF UN-TRANSLATED KEYCODE (IF NEEDED)
              4 EXTRA STACK LEVELS USED (MAX)
              (0 EXTRA STACK LEVELS USED - IN DOSF62N ($KEYCHAR) ENTRY)
.
      CALL      BLINKDE     BLINK THE CURSOR AT (DE)
DOSF62N  MLA      *SEKBS1     GET KEYBOARD STATUS 1 INTO (A)
        ND      SEKBRDY      EXIT IF KEYBOARD NOT READY

```

154,K 175552 053
155,K
156,K 175553 257
157,K 175554 370
158,K 175555 066 152 367
159,K 175560 176 044 177
160,K 175563 056 334
161,K 175565 307
162,K 175566 115 265
163,K 175570 007
164,K
165,K 175571
166,K
167,K
168,K
169,K
170,K
171,K
172,K
173,K
174,K
175,K
176,K
177,K
178,K
179,K
180,K
181,K
182,K
183,K
184,K
185,K
186,K
187,K
188,K
189,K 175571 106 312 374
190,K 175574 043
191,K
192,K 175575 371
193,K 175576 007

RTZ

XRM

ELSE RESET READY BIT

LMA

MLL

SEKBCB

GET THE CHARACTER

NDL

0177

STRIP OFF THE SIGN BIT

LH

SEKTRAN>8

GENERATING INDEX INTO TRANSLATE TABLE

LAM

GET TRANSLATED CODE

ORHH

INSURE ZERO FLAG IS FALSE

RET

*
DOSF63

DOS FUNCTION: 6 SUBFUNCTION: 3

- WRITE A BYTE TO THE SCREEN -

ENTRY: B = BYTE TO BE WRITTEN

D = HORIZONTAL CURSOR POSITION

E = VERTICAL CURSOR POSITION

EXITS: IF D OR E OUT OF RANGE:

CARRY TRUE

A = SCRATCHED

IF D AND E WITHIN RANGE:

A = SCRATCHED

HL = BUFFER ADDRESS OF BYTE WRITTEN

1 EXTRA STACK LEVELS USED (MAX)

NOTE: THIS ROUTINE CAN OVER-WRITE A BLINKING CURSOR

BUT THE BLINK ROUTINE IS WRITTEN TO ALLOW IT BY NOT

RESTORING THE CHARACTER BEHIND THE CURSOR IF THE

CURSOR IS NOT THERE ANY MORE.

NOT CHECKED BECAUSE FASTER BY ELIMINATING CALL TO CURSES.

CALL

DSPCBL

COMPUTE BUFFER LOCATION

RTC

EXIT IF OUT OF RANGE

LMB

ELSE STORE THE BYTE

RET

AND EXIT FALSE CARRY

194,K
195,K 175577
196,K
197,K
198,K
199,K
200,K
201,K
202,K
203,K
204,K
205,K 175577 046 364
206,K 175601 036 000
207,K 175603 007
208,K
209,K 175604
210,K
211,K
212,K
213,K
214,K
215,K
216,K
217,K
218,K
219,K 175604 046 013
220,K 175606 036 000
221,K 175610 007
222,K
223,K 175611
224,K
225,K
226,K
227,K
228,K
229,K
230,K
231,K
232,K
233,K
234,K
235,K
236,K 175611 066 242 056 357 307
237,K 175616 064 001
238,K 175620 370
239,K

*
DOSF64
.
DOS FUNCTION: 6 SUBFUNCTION: 4
.
- RETURN HOME UP POSITION IN (DE) -
.
EXITS: D = HOME UP HORIZONTAL CURSOR POSITION
E = HOME UP VERTICAL CURSOR POSITION
0 EXTRA STACK LEVELS USED (MAX)
.
LE BL+1=MAXLIN
LD LC
RET
*
DOSF65
.
DOS FUNCTION: 6 SUBFUNCTION: 5
.
- RETURN HOME DOWN POSITION IN (DE) -
.
EXITS: D = HOME DOWN HORIZONTAL CURSOR POSITION
E = HOME DOWN VERTICAL CURSOR POSITION
0 EXTRA STACK LEVELS USED (MAX)
.
LE BL
LD LC
RET
*
DOSF66
.
DOS FUNCTION: 6 SUBFUNCTION: 6
.
- TURN THE CURSOR ON -
.
ENTRY: D = HORIZONTAL CURSOR POSITION
E = VERTICAL CURSOR POSITION
.
EXITS: A,B,C,H,L = SCRATCHED
IF D OR E OUT OF RANGE: CURSOR "ON" BUT NOT FLASHING
3 EXTRA STACK LEVELS USED (MAX)
.
MLA *SECFLAGS SET THE CURSOR ON/OFF SWITCH TO ON
OR CURONOFF
LMA
.
FALLS THROUGH INTO BLINKDE

240,K
241,K 175621
242,K
243,K
244,K
245,K
246,K
247,K 175621 066 245 056 357
248,K 175625 111 047
249,K 175627 062 254
250,K 175631 111 253
251,K 175633 062 261
252,K 175635 022 070
253,K 175637 112 237 374
254,K 175642 022 060
255,K
256,K 175644
257,K
258,K
259,K
260,K
261,K
262,K
263,K
264,K
265,K
266,K
267,K
268,K
269,K
270,K 175644 066 242 056 357 307
271,K 175651 044 001
272,K 175653 053
273,K 175654 307
274,K 175655 066 153
275,K 175657 257
276,K 175660 044 010
277,K 175662 053
278,K
279,K 175663 066 242
280,K 175665 257
281,K 175666 370
282,K 175667 066 245
283,K 175671 047
284,K 175672 106 312 374
285,K 175675 043
286,K
287,K 175676 022 070
288,K 175700 076 357
289,K 175702 105 242
290,K 175704 044 010
291,K 175706 105 244

*
BLINKDE

. IF THE CURSOR POSITION GIVEN IN (DE) IS DIFFERENT FROM THE OLD POSITION,
. SUSPEND THE CURSOR AND UPDATE THE OLD CURSOR POSITION TO THE NEW.
. 3 EXTRA STACK LEVELS USED (MAX)

HL SECPOS BC = OLD CURSOR POSITION
DL BC,HL
XREC SEE IF BC = DE
XROB
ORBC
PUSH XA SAVE XA IN CASE CURSES IS CALLED
CFZ CURSES SUSPEND CURSOR IF BC .NE. DE
POP XA RESTORE XA

*
BLINK

. IF THE CURSOR ON/OFF SWITCH IS ON AND THE LINE FREQUENCY COUNTER BIT
. USED FOR FLASHING IS DIFFERENT FROM THE CURSOR "IN BUFF" BIT,
. UPDATE THE CURSOR "IN BUFF" BIT AND THEN IF THE CURSOR POSITION
. IS WITHIN THE RANGE OF THE SCREEN, EITHER RESTORE THE HIDDEN
. CHARACTER INTO THE BUFFER IF THE "IN BUFF" BIT IS NOW A ZERO OR SAVE
. THE BUFFER LOCATION IN THE HIDDEN CHARACTER AND PUT THE CURSOR
. CHARACTER IN THE BUFFER IF THE "IN BUFF" BIT IS A ONE.
. IF ON RESTORING THE HIDDEN CHARACTER IT IS NOTED THAT THE CURSOR
. CHARACTER IS NOT IN THE BUFFER (OVERWRITTEN) THEN THE CHARACTER IS
. NOT RESTORED (ALLOWING WRITING IN THE BUFFER ON TOP OF THE CURSOR).
. 1 EXTRA STACK LEVELS USED (MAX)

MLA *SECFLAGS EXIT IF THE CURSOR IS OFF
ND CURONOFF
RTZ
LAM EXIT IF THE "OFF" BIT EQUALS THE
LL SELFREQ LINE FREQUENCY BIT
XRM
ND CURINBUF
RTZ
LL SECFLAGS UPDATE THE "IN BUF" BIT
XRM
LMA
LL SECPOS GET THE CURSOR POSITION
DL DE,HL
CALL DSPCBL GET THE MEMORY ADDRESS IN THE BUFFER
RTC EXIT IF THE CURSOR IS OFF THE SCREEN
PUSH XA SAVE THE X-REGISTER
LX SECFLAGS>8 SET UP THE PAGE REGISTER
PL A,SECFLAGS
ND CURINBUF BRANCH ON CURSOR POSITION (IN/OUT BUFFER)
PL A,SECCHAR (GET THE CURSOR CHARACTER)

292.K	175710	150 322 373	JTZ	BLINKOUT	RESTORE CHARACTER IF THE CURSOR WENT OUT
293.K					
294.K	175713	327	LCM		ELSE SAVE THE NEW HIDDEN CHARACTER
295.K	175714	126 243	PS	C,SECHIDE	AND PUT THE CURSOR CHARACTER
296.K	175716	370	LMA		INTO THE DISPLAY BUFFER
297.K	175717	022 060	POP	XA	RESTORE THE XA REGISTERS
298.K	175721	007	RET		
299.K					
300.K	175722	124 243	BLINKOUT PL	C,SECHIDE	(GET THE HIDDEN CHARACTER)
301.K	175724	277	CPM		IS THE CURSOR STILL IN MEMORY
302.K	175725	022 060	POP	XA	
303.K	175727	013	RFZ		NO, IT WAS OVERWRITTEN
304.K	175730	372	LMC		YES, RESTORE THE HIDDEN CHARACTER
305.K	175731	007	RET		

306,K				+		
307,K				.		
308,K				.		** MOVED HERE TO SAVE 3 BYTES ***
309,K	175732	036	000	DISPNEWL	LD	LC
310,K	175734	174	004 001		AD	1
311,K	175737	174	074 014		CPE	BL+1
312,K	175742	013			RFZ	
313,K				.		
314,K	175743	046	013		LE	BL
315,K					JMP	DOSF67
316,K				.		ELSE SET VERT POSITION TO BOTTOM
317,K	175745			.		OF SCREEN, ROLL UP, AND RETURN
318,K				+		
319,K				DOSF67		
320,K				.		
321,K				.	DOS FUNCTION: 6	SUBFUNCTION: 7
322,K				.		
323,K				.		
324,K				.		
325,K				.		
326,K				.		
327,K				.		
328,K				.		
329,K				.		
330,K				.		
331,K				.		
332,K				.		
333,K				.		
334,K				.		
335,K	175745	106	225 374	.		
336,K	175750	106	312 374	.		
337,K	175753	043		.		
338,K				.		
339,K	175754	115	164 156	.		
340,K	175757	115	166 236	.		
341,K				.		
342,K	175762	250		.		
343,K	175763	106	114 374	.		
344,K				.		
345,K	175766	046	156 036 357	.		
346,K	175772	353		.		
347,K	175773	066	160	.		
348,K	175775	026	060	.		
349,K	175777	250		.		
350,K	176000	310		.		
351,K	176001	021		.		
352,K				.		
353,K	176002	250		.		
354,K	176003	007		.		

355,K
356,K 176004
357,K
358,K
359,K
360,K
361,K 176004 046 000 036 320
362,K 176010 066 156 056 357
363,K 176014 006 120
364,K
365,K 176016 027
366,K 176017 174 017
367,K 176021 117 015
368,K 176023 176 074 236
369,K 176026 110 016 374
370,K
371,K 176031 250
372,K 176032 066 241 370
373,K
374,K 176035 066 242 370
375,K
376,K 176040 066 244 370
377,K
378,K 176043 106 177 373
379,K
380,K
381,K 176046
382,K
383,K
384,K
385,K
386,K
387,K
388,K
389,K
390,K
391,K
392,K
393,K
394,K
395,K
396,K
397,K
398,K
399,K 176046 106 225 374
400,K 176051 016 040
401,K
402,K 176053 106 312 374
403,K 176056 043
404,K
405,K 176057 303
406,K 176060 331

```

+
DSPINIT
.
. INITIALIZE ALL DISPLAY POINTERS AND CONTROL INFORMATION
. 3 EXTRA STACK LEVELS USED (MAX)
.
      DE      SDSPBUF      POINT TO THE SCREEN BUFFER
      HL      SEDLN01      POINT TO THE POINTER LIST
      LA      MAXPOS       LENGTH OF EACH LINE
.
DSPINITL DS      DE,HL      SET UP A LINE POINTER
          INCP     DE,A      BUMP THE POINTER VALUE
          INCP     HL,2      BUMP THE POINTER ADDRESS
          CPL      SEDLN01+(2*MAXLIN) GO UNTIL ALL LINES DONE
          JFZ      DSPINITL
.
          XRA              ZERO THE DISPLAY ROUTINE FLAGS
          MSA              SEDOPTS+1      USED BY THE DOS FUNCTIONS
.
          MSA              SECFLAGS      SET CURSOR FLAGS TO SUSPENDED
          LA               AND NOT IN THE BUFFER
          MSA              INITIALIZE THE CHARACTER (ASSUMED ZERO)
.                               USED FOR THE CURSOR
          CALL     DOSF64      GET HOME UP POSITION IN (DE)
          JUMP     DOSF68      ERASE THE ENTIRE SCREEN AND EXIT
.
*
DOSF68
.
. DOS FUNCTION: 6 SUBFUNCTION: 8
.
. - ERASE FROM THE CURSOR THRU THE END OF FRAME -
.
. ENTRY: D = HORIZONTAL CURSOR POSITION
.        E = VERTICAL CURSOR POSITION
.
. EXITS: ALL REGISTERS PRESERVED
.        IF D OR E OUT OF RANGE:
.            CARRY TRUE
.            NO LINES ERASED
.        IF D AND E BOTH WITHIN RANGE:
.            CARRY FALSE
.            LINE(S) ERASED
.        3 EXTRA STACK LEVELS USED (MAX)
.
          CALL     CURSESRS      SAVE REGISTERS AND SUSPEND CURSOR
          LB              ' '      STANDARD ERASE CHARACTER
.
DOSF68LP CALL     DSPCBL      COMPUTE THE BUFFER POSITION
          RTC              RESTORE REGS AND EXIT IF DE OUT OF RANGE
.
          LAD              BLANK TO THE END OF THAT LINE
          LDB              (SAVE THE B-REGISTER)

```

407,K 176061 174 070
408,K 176063 106 116 374
409,K 176066 174 060
410,K 176070 313
411,K 176071 036 000
412,K 176073 174 004 001
413,K 176076 174 074 014
414,K 176101 110 053 374
415,K
416,K 176104 007
417,K
418,K 176105
419,K
420,K
421,K
422,K
423,K
424,K
425,K
426,K
427,K
428,K
429,K
430,K 176105 106 225 374
431,K 176110 106 312 374
432,K 176113 303
433,K
434,K
435,K 176114
436,K
437,K
438,K
439,K
440,K
441,K
442,K
443,K
444,K
445,K
446,K 176114 016 040
447,K 176116 026 117
448,K 176120 062 220
449,K 176122 105 241
450,K 176124 044 200
451,K 176126 261
452,K 176127 370
453,K 176130 062 262
454,K 176132 053
455,K
456,K 176133 346
457,K 176134 335
458,K 176135 174 015

```

      PUSH      DE
      CALL      BLANKITB
      POP       DE
      LBD
      LD         LC          (RESTORE THE B-REGISTER)
      ADE        1          RESET THE HORIZONTAL POSITION
      CPE        BL+1       INCRMENT THE VERTICAL LINE POSITION
      JFZ        DOSF68LP   ERASE UNTIL BOTTOM LINE REACHED

      RET

DOSF69
      DOS FUNCTION: 6  SUBFUNCTION: 9
      - ERASE FROM THE CURSOR THRU THE END OF LINE -
      ENTRY: D = HORIZONTAL CURSOR POSITION
             E = VERTICAL CURSOR POSITION
      EXITS: (SEE "DOSF68" ABOVE)
             3 EXTRA STACK LEVELS USED (MAX)

      CALL      CURSESRS    SAVE REGISTERS AND SUSPEND CURSOR
      CALL      DSPCBL      POINT TO LINE TO BE BLANKED
      LAD
      JMP       BLANKIT     ERASE TO THE END OF THE LINE

      BLANKIT
      BLANK A LINE WITH EITHER NORMAL OR INVERTED VIDEO BLANKS
      ENTRY    A          = HORZ CURSOR POSITION
              B          = BLANKING CHARACTER (IF ENTER AT "BLANKITB")
              HL         = ADDRESS OF THE LINE
              X          = SEDOPTS>8
              SEDOPTS    = INVERTED VIDEO OPTION IN B7
              0 EXTRA STACK LEVELS USED

      BLANKITB  LB         1 1          BLANK LINE WITH STANDARD BLANK CODE
                LC         MAXPOS-1    COMPUTE HOW MANY CHARACTERS TO BLANK
                SUAC
                PL         A,SEDOPTS+1  GET THE INVERTED VIDEO OPTION FLAG
                ND         $OI
                ORB
                LMA
                ORCC
                RTZ

                LEL
                LDH
                INCP      DE          DE => NEXT CHARACTER
    
```

459,K 176137 250
460,K 176140 310
461,K 176141 021
462,K 176142 250
463,K 176143 007
464,K
465,K 176144
466,K
467,K
468,K
469,K
470,K
471,K
472,K
473,K
474,K
475,K
476,K
477,K 176144 106 225 374
478,K 176147 106 312 374
479,K 176152 043
480,K
481,K 176153 115 164 234
482,K 176156 115 166 154
483,K
484,K 176161 250
485,K 176162 106 114 374
486,K
487,K 176165 046 235 036 357
488,K 176171 353
489,K 176172 066 233
490,K 176174 026 060
491,K 176176 250
492,K 176177 310
493,K 176200 111 021
494,K
495,K 176202 250
496,K 176203 007
497,K
498,K 176204
499,K
500,K
501,K
502,K
503,K
504,K
505,K
506,K
507,K
508,K
509,K
510,K

XRA
LBA
BT
XRA
RET
ELSE DO THE BLOCK TRANSFER
TO CLEAR THE REST OF THE LINE
EXIT FALSE CARRY

*
DOSF610

. DOS FUNCTION: 6 SUBFUNCTION: 10

. - ROLL THE SCREEN DOWN ONE LINE -

. ENTRY: D = HORIZONTAL CURSOR POSITION
. E = VERTICAL CURSOR POSITION

. EXITS: (SEE "DOSF67" ABOVE)
. 3 EXTRA STACK LEVELS USED (MAX)

. CALL CURSESRS SAVE REGISTERS AND SUSPEND CURSOR
CALL DSPCBL SEE IF DE WITHIN RANGE
RTC RESTORE REGS AND EXIT IF NOT

. DPL HL,SEDLBOT PUT THE ADDRESS OF THE BOTTOM LINE
DPS HL,SEDNUL IN THE POINTER ABOVE THE TOP LINE

. XRA
CALL BLANKIT BLANK OUT THE WHOLE LINE

. DE SEDLBOT+1 ROLL DOWN THE LINE POINTERS

. LHD
LL SEDLBOT+2+1
LC MAXLIN*2 (ALL THE LINES)

. XRA
LBA
BTR

. XRA SET CARRY FALSE
RET RESTORE THE REGISTERS AND EXIT

*
DOSF611

. DOS FUNCTION: 6 SUBFUNCTION: 11

. - TURN THE CURSOR OFF -

. ENTRY: D = HORIZONTAL CURSOR POSITION
. E = VERTICAL CURSOR POSITION

. EXITS: ALL REGISTERS PRESERVED
. 3 EXTRA STACK LEVELS USED (MAX)
. A-REG CHANGED IF DOSF611N ENTRY USED, CURSOR MUST BE SUSPENDED

511,K 176204 106 225 374
512,K 176207
513,K 176207 105 242
514,K 176211 044 376
515,K 176213 107 242
516,K 176215 007
517,K
518,K 176216
519,K
520,K
521,K
522,K
523,K
524,K
525,K
526,K
527,K
528,K
529,K
530,K
531,K
532,K
533,K
534,K 176216 026 150 016 357
535,K 176222 062 047
536,K 176224 007

CALL CURSESRS SAVE REGISTERS AND SUSPEND CURSOR
DOSF611N PL A,SECFLAGS SET THE CURSOR ON/OFF BIT TO "OFF"
ND -1-CURONOFF
PS A,SECFLAGS
RET RESTORE THE REGISTERS AND EXIT
*
DOSF612
· DOS FUNCTION: 6 SUBFUNCTION: 12
· - RETURN FUNCTION KEY AND OTHER STATUS -
·
· EXITS: ALL REGISTERS PRESERVED
· BC EXCEPTED, THEY CONTAIN THE STATUS BITS
· R = (X, X, X, X, SEKBOWN, SEKBDY, SEKBDKY, SEDSPKY)
· C = (SEINTKY, SEATTKY, SERSTKY, SEFUNC5, 4, 3, 2, 1)
·
· 0 EXTRA STACK LEVELS USED (MAX)
·
· WARNING: CHANGING OF LOCATIONS SEKBS2 OR SEKBS1 WITHOUT DUE REGARD FOR EFFECT
· CAN CAUSE LOSS OF CHARACTERS, FUNCTION KEYS AND AUTO RESTARTS!!!
·
· BC SEKBS2
· DL BC,BC GET THE STATUS BYTES
· RET

539,K
540,K 176225
541,K
542,K
543,K
544,K
545,K
546,K 176225 051 271 335
547,K 176230 055
548,K 176231 060
549,K 176232 060
550,K 176233 051 303 374
551,K 176236 070
552,K
553,K 176237
554,K
555,K
556,K
557,K
558,K
559,K
560,K
561,K
562,K
563,K
564,K
565,K
566,K
567,K
568,K 176237 076 357
569,K 176241 115 164 245
570,K 176244 113 146 245
571,K 176247 105 242
572,K 176251 044 010
573,K 176253 053
574,K
575,K
576,K
577,K
578,K 176254 105 242
579,K 176256 044 367
580,K 176260 107 242
581,K 176262 346
582,K 176263 335
583,K 176264 106 312 374
584,K 176267 113 144 245
585,K 176272 043
586,K
587,K 176273 105 244
588,K 176275 277
589,K 176276 013
590,K 176277 105 243

*
CURSESRS

. SAVE THE REGISTERS AND CAUSE A "RETURN" AFTER THE CALL TO THIS ROUTINE
. TO TRANSFER CONTROL TO "DSPRESTR". THEN SUSPEND THE CURSOR.
. 3 EXTRA STACK LEVELS USED (MAX)

PUSH	DSPREGSA+7	SAVE THE REGISTERS IN THE
REGS		AREA FOR THE DISPLAY
POP	HL	GET RID OF "DSPREGSA" ADDRESS ON STACK
POP	HL	PUT "DSPRESTR" ON STACK BELOW
PUSH	DSPRESTR	THE RETURN ADDRESS
PUSH	HL	

*
CURSES

. SAVE THE NEW CURSOR POSITION GIVEN IN (DE) IN "SECPOS". THEN, IF
. THE CURSOR CHARACTER IS IN THE BUFFER, SUSPEND THE CURSOR
. BY PLACING THE CHARACTER IN "SECHIDE" BACK INTO THE BUFFER
. AS DIRECTED BY THE OLD CURSOR POSITION BEFORE THIS ROUTINE
. WAS ENTERED.

ENTRY: D = HORIZONTAL CURSOR POSITION
E = VERTICAL CURSOR POSITION

EXITS: HL,A = SCRATCHED
X => SECFLAGS PAGE
1 EXTRA STACK LEVELS USED (MAX)

LX	SECFLAGS>8	
DPL	HL,SECPOS	SAVE THE OLD CURSOR POSITION IN HL
DPS	DE,SECPOS	UPDATE THE CURSOR POSITION TO THAT IN DE
PL	A,SECFLAGS	EXIT IF THE CURSOR CHARACTER IS
ND	CURINBUF	NOT CURRENTLY IN THE BUFFER
RTZ		(THE "CURIN BUFF" BIT IS A ZERO)

*
. CLEAR THE CURSOR ON BIT AND STORE THE HIDDEN CHARACTER IN THE
. CURSOR POSITION GIVEN IN (HL)

PL	A,SECFLAGS	ELSE CLEAR THE CURSOR IN BUFF BIT
ND	=1-CURINBUF	
PS	A,SECFLAGS	
LEL		HL => WHERE THE CURSOR WAS
LDH		

CALL	DSPCBL	
DPL	DE,SECPOS	RESTORE THE NEW CURSOR POSITION IN DE
RTC		EXIT IF CURSOR WAS OFF THE SCREEN

PL	A,SECCHAR	CHECK IF THE CURSOR IS STILL IN MEMORY
CPM		IF NOT, DON'T RESTORE THE HIDDEN CHAR
RFZ		
PL	A,SECHIDE	ELSE, STORE THE HIDDEN CHARACTER

591,K	176301	370		LMA		WHERE THE CURSOR WAS
592,K	176302	007		RET		AND EXIT
593,K				*		
594,K	176303	066	271 056 335	DSPRESTR	HL	DSPREGSA+7
595,K	176307	111	055		REGL	RESTORE THE REGISTERS FROM THE
596,K	176311	007			RET	SAVE AREA FOR THE DISPLAY
597,K						ROUTINES.
598,K	176312			*		
599,K				DSPCBL		
600,K				.		
601,K				.		COMPUTE DISPLAY BUFFER LOCATION
602,K				.		
603,K				.		ENTRY: D = HORIZONTAL CURSOR POSITION
604,K				.		E = VERTICAL CURSOR POSITION
605,K				.		
606,K				.		EXITS: IF D AND E BOTH WITHIN RANGE:
607,K				.		CARRY FALSE
608,K				.		HL = MEMORY LOCATION IN THE SCREEN BUFFER
609,K				.		A = ZERO
610,K				.		IF D OR E NOT WITHIN RANGE:
611,K				.		CARRY TRUE
612,K				.		A = SCRATCHED
613,K				.		0 EXTRA STACK LEVELS USED (MAX)
614,K	176312	303		.		
615,K	176313	004	260	LAD		EXIT TRUE CARRY IF D OUT OF RANGE
616,K	176315	043		AD	-MAXPOS	
617,K				RTC		
618,K	176316	304		.		
619,K	176317	024	014	LAE		EXIT TRUE CARRY IF E OUT OF RANGE
620,K	176321	074	350	SU	BL+1	MAKE A = -MAXLIN THRU -1
621,K	176323	043		CP	-MAXLIN	
622,K				RTC		
623,K	176324	200		.		
624,K	176325	066	236 056 356	ADA		DOUBLE UNBIASED E VALUE TO GET INDEX
625,K	176331	017		HL	SEDNUB=256	INTO THE LIST OF LINE POINTERS
626,K	176332	057		INCP	HL,A	(256 CORRECTS FOR SINGLE BYTE INCP)
627,K	176333	303		DL	HL,HL	GET POINTER TO BEGINNING OF LINE
628,K	176334	017		LAD		INDEX INTO THE SELECTED LINE
629,K	176335	250		INCP	HL,A	
630,K	176336	007		XRA		EXIT FALSE CARRY
				RET		

633,K	*			
634,K	.	DISPLAY A STRING ROUTINE		
635,K	.			
636,K	.	CONTROL CHARACTERS:		
637,K	.			
638,K 000003	\$ES*	EQU	003	END OF DISPLAY STRING
639,K 000007	\$BP*	EQU	007	BEEP
640,K 000011	\$H*	EQU	011	NEW DISPLAY COLUMN FOLLOWS
641,K 000013	\$V*	EQU	013	NEW DISPLAY ROW FOLLOWS
642,K 000015	\$EL*	EQU	015	ADVANCE TO NEW LINE AND TERMINATE STRING
643,K 000021	\$EEOF*	EQU	021	ERASE TO END OF FRAME
644,K 000022	\$EEOI*	EQU	022	ERASE TO END OF LINE
645,K 000023	\$RU*	EQU	023	ROLL UP
646,K 000024	\$RD*	EQU	024	ROLL DOWN
647,K 000033	\$F*	EQU	033	FORCE DISPLAY OF NEXT CHARACTER
648,K	.			
649,K 000203	\$NS*	EQU	0203	NEW STRING ADDRESS FOLLOWS (LSB,MSB)
650,K 000207	\$CK*	EQU	0207	CLICK
651,K 000211	\$HA*	EQU	0211	DISPLAY COLUMN ADJUSTMENT FOLLOWS
652,K 000213	\$VA*	EQU	0213	DISPLAY ROW ADJUSTMENT FOLLOWS
653,K 000215	\$NL*	EQU	0215	ADVANCE TO NEW LINE
654,K	.	EQU	0221	* NOT USED *
655,K	.	EQU	0222	* NOT USED *
656,K 000223	\$HU*	EQU	0223	HOME UP (UPPER LEFT-HAND CORNER)
657,K 000224	\$HD*	EQU	0224	HOME DOWN (LOWER LEFT-HAND CORNER)
658,K 000233	\$O*	EQU	0233	NEW OPTIONS FOLLOW
659,K	.			
660,K 000200	\$OI*	EQU	B7	INVERTED VIDEO OPTION
661,K 000100	\$OS*	EQU	B6	SKIP BLANKS OPTION
662,K 000040	\$OW*	EQU	B5	INHIBIT WAIT ON DISPLAY KEY OPTION

```

663,K
664,K
665,K
666,K
667,K
668,K
669,K
670,K
671,K
672,K
673,K
674,K
675,K
676,K
677,K
678,K
679,K
680,K
681,K
682,K
683,K
684,K 176337 016 140
685,K
686,K 176341 022 070
687,K 176343 070
688,K 176344 106 237 374
689,K 176347 111 126 240
690,K
691,K 176352 060
692,K 176353 006 377
693,K 176355 107 246
694,K
695,K 176357
696,K
697,K
698,K
699,K 176357 307
700,K 176360 015
701,K
702,K 176361 310
703,K 176362 124 241
704,K
705,K 176364 074 033
706,K 176366 150 062 375
707,K 176371 070
708,K 176372 140 007 375
709,K 176375 074 203
710,K 176377 140 065 375
711,K 176402 074 234
712,K 176404 100 065 375
713,K
714,K 176407 066 152 056 375

```

```

*
. THE DISPLAY ROUTINE ENTRY POINTS
.
. "DISPDOS" INITIALIZES THE OPTIONS IN THE B-REGISTER TO
.   A DOS-COMPATIBLE SET
. "DISPLAY" STORES THE OPTIONS IN THE B-REGISTER IN "SEDOPTS"
.
. ENTRY: B = OPTIONS (UNLESS ENTERING AT DISPDOS)
.         D = HORIZONTAL CURSOR POSITION (0 THRU 79)
.         E = VERTICAL CURSOR POSITION (-12 THRU 11)
.         HL = STRING ADDRESS
.
. EXITS: A = ENTRY VALUE
.         B = LAST OPTION VALUE SET
.         C = ENTRY VALUE
.         DE = CURSOR POSITION AFTER LAST CHARACTER DISPLAYED
.             OR CURSOR POSITION WHEN LAST CONTROL EXECUTED
.         HL = ADDRESS OF BYTE AFTER STRING TERMINATOR
.         6 EXTRA STACK LEVELS USED (MAX)
.           EXCLUDING DISPLAY WAIT VECTOR OVERHEAD (IF USED)
.
. DISPDOS  LB          $OS+$OW          SKIP BLANKS AND INHIBIT DSP KEY WAIT
.
. DISPLAY  PUSH        XA              SAVE THE X-REGISTER
.          PUSH        HL              SAVE THE STRING POINTER
.          CALL        CURSES          MAKE SURE THE CURSOR IS SUSPENDED
.          DPS         BC,SEDOPTS      SAVE THE BC REGISTER PAIR IN 'SEDOPTS'
.
*
. DISPCPRP POP         HL              RESTORE THE STRING POINTER
. DISPCPSL LA          =1              CLEAR THE PHYSICAL SCREEN POINTER
.          PS          A,SECPOS+1      MSB TO ALL ONES (CAN BE IN ROM!)
.
*
. DISPLANC
.
. DISPLAY THE NEXT CHARACTER IN THE STRING POINTED TO BY (HL)
.
.          LAM
.          INCP        HL              GET THE NEXT STRING CHARACTER
.                                         BUMP THE STRING POINTER
.
.          LBA
.          PL          C,SEDOPTS+1     SAVE THE CHARACTER IN THE B-REG
.                                         C = DISPLAY OPTION BITS
.
.          CP          SF              FALL THROUGH IF THE CHARACTER
.          JNZ         DISPFORC        (SPECIAL HANDLING)
.          PUSH        HL              (SAVE THE STRING POINTER)
.          JTC         DISPFUNC        CANNOT BE A CONTROL
.          CP          $NS
.          JTC         DISPOUT
.          CP          $D+1
.          JFC         DISPOUT
.
. DISPFUNC HL          DISPFTAB-2     SEE IF FUNCTION IN TABLE

```


715,K 176413 117 015
716,K 176415 307
717,K 176416 260
718,K 176417 150 065 375
719,K
720,K 176422 015
721,K 176423 271
722,K 176424 110 013 375
723,K
724,K 176427 302
725,K 176430 044 040
726,K 176432 110 047 375
727,K
728,K 176435 106 105 357
729,K
730,K 176440 105 151
731,K 176442 044 001
732,K 176444 110 035 375
733,K
734,K 176447 062 060
735,K 176451 051 353 374
736,K 176454 057
737,K 176455 070
738,K 176456 362
739,K 176457 351
740,K 176460 250
741,K 176461 007
742,K
743,K 176462 317
744,K 176463 015
745,K 176464 070
746,K
747,K
748,K 176465 302
749,K 176466 044 200
750,K 176470 111 250
751,K 176472 115 164 245
752,K 176475 305
753,K 176476 004 001
754,K 176500 142 312 374
755,K 176503 140 146 375
756,K
757,K 176506 006 177
758,K 176510 241
759,K 176511 074 040
760,K 176513 110 124 375
761,K
762,K 176516 302
763,K 176517 044 100
764,K 176521 110 125 375
765,K
766,K 176524 371

DISPFCLP INCP HL,2
LAM
ORA
JTZ DISPOUT WRITE CHAR IF TERMINATOR REACHED
.
INCP HL HL => FUNCTION EXECUTION ADDRESS
CPB SEE IF CONTROL CHARACTER DESIRED
JFZ DISPFCLP KEEP SCANNING IF NOT
.
LAC
ND SDW SEE IF THE "INHIBIT WAIT BEFORE CONTROL"
JFZ DISPDOCC OPTION BIT IS SET
JUST DO CONTROL FUNCTION IF IT IS
.
DISPDSPW CALL SVDISPWS CALL THE DISPLAY WAIT ROUTINE
.
PL A,SEKBS1 (WILL ALWAYS DO IT ONCE IF ENABLED!!)
ND SEOSPKY ELSE SEE IF THE DISPLAY KEY IS DOWN
JFZ DISPDSPW WAIT ON IT IF IT IS
.
DISPDOCC POP BC PUT STRING POINTER INTO BC FOR A WHILE
PUSH DISPCPSL PUT NEXT CHAR RETURN ADR ON THE STACK
DL HL,HL GET THE CONTROL ROUTINE ADDRESS
PUSH HL AND PUT THAT ON THE STACK
LLC MAKE HL => NEXT STRING CHARACTER
LMB
XRA
RET
*
DISPFORC LBM
INCP HL FORCE THE NEXT CHARACTER
PUSH HL TO BE DISPLAYED REGARDLESS OF ITS
JUMP HL VALUE
NOTE: SPECIAL HANDLING OF THIS SF CODE
*
DISPOUT LAC
ND B7 ISOLATE THE VIDEO INVERSION OPTION BIT
XRAB AND INVERT THE SIGN BIT OF THE
DPL HL,SECPDS CHARACTER TO BE WRITTEN IF SET
LAH GET THE PHYSICAL SCREEN POSITION
AD 1 COMPUTE THE PHYSICAL LOCATION
CTC DSPCBL IF IT IS NOT KNOWN (IN ROM!)
JTC DISPOFFS CATCH OFF THE SCREEN
.
LA 0177 CATCH WRITING SPACES
NDB
CP ' ' JUST WRITE IF NOT
JFZ DISPWRIT
.
LAC
ND B6 ELSE SEE IF SPACES ARE TO BE SKIPPED
JFZ DISPSKIP DON'T WRITE IF SO
.
DISPWRIT LMB ELSE STORE THE BYTE ON THE SCREEN

767,K 176525 015
768,K 176526 115 166 245
769,K 176531 060
770,K 176532 113 004 001
771,K 176535 113 074 120
772,K 176540 140 357 374
773,K
774,K 176543 104 353 374
775,K
776,K 176546 113 004 001
777,K 176551 104 352 374
778,K
779,K 176554
780,K
781,K
782,K
783,K 176554 011 254 375
784,K 176557 013 261 375
785,K 176562 211 253 375
786,K 176565 213 260 375
787,K 176570 215 332 373
788,K 176573 203 240 375
789,K 176576 223 177 373
790,K 176601 224 204 373
791,K
792,K 176604 233 242 375
793,K 176607 023 345 373
794,K 176612 024 144 374
795,K 176615 022 105 374
796,K 176620 021 046 374
797,K 176623 007 247 375
798,K 176626 207 251 375
799,K 176631 015 265 375
800,K 176634 003 270 375
801,K 176637 000
802,K
803,K 176640 057
804,K 176641 007
805,K
806,K 176642 307
807,K 176643 015
808,K 176644 107 241
809,K 176646 007

DISPSKIP INCP HL INCREMENT THE PHYSICAL SCREEN POSITION
DPS HL,SECPOS SAVE THE PHYSICAL SCREEN POSITION
POP HL RESTORE THE STRING POINTER
ADD 1 INCREMENT THE CURSOR POSITION
CPD MAXPOS ON TO THE NEXT STRING CHARACTER
JTC DISPLANC IF CURSOR STILL ON THE SCREEN
*
JMP DISPCPSL ELSE CLEAR THE PHYSICAL SCREEN LOCATION
*
DISPOFFS ADD 1 INCREMENT THE CURSOR POSITION AND
JMP DISPCPRP ON TO THE NEXT STRING CHARACTER
*
DISPFTAB
*
* DISPLAY FUNCTION TABLE
*
DC SH,*DISPHCOL
DC SV,*DISPVCOL
DC SHA,*DISPHADJ
DC SVA,*DISPVADJ
DC SNL,*DISPNEWL
DC SNS,*DISPGNSA
DC SHU,*DOSF64
DC SHD,*DOSF65
* DC SF,*DISPFORC (SPECIAL HANDLING)
DC SN,*DISPGOPT
DC SRU,*DOSF67
DC SRD,*DOSF610
DC SEEDL,*DOSF69
DC SEEDF,*DOSF68
DC SBP,*DISPBEEP
DC SCK,*DISPCLIK
DC SEL,*DISPENDL
DC SES,*DISPEXIT
DC 0 TERMINATOR
*
DISPGNSA DL HL,HL GET NEW STRING ADDRESS
RET
*
DISPGOPT LAM
INCP HL GET THE NEW OPTIONS
PS A,SEDOPTS+1 INCREMENT THE STRING POINTER
RET STORE THE NEW OPTIONS

810,K						
811,K	176647	151		*	DISPBEEP	EX BEEP
812,K	176650	007				RET
813,K				*		
814,K	176651	153			DISPCLIK	EX CLICK
815,K	176652	007				RET
816,K				*		
817,K	176653	303			DISPHADJ	LAD CAUSE HORIZONTAL ADJUSTMENT
818,K						WHEN ENTERED HERE
819,K	176654	207		*	DISPHCOL	ADM A = ZERO WHEN ENTERED HERE
820,K	176655	330				LDA SO D = STRING VALUE
821,K	176656	015				INCP HL
822,K	176657	007				RET
823,K				*		
824,K	176660	304			DISPVADJ	LAE CAUSE VERTICAL ADJUSTMENT
825,K						WHEN ENTERED HERE
826,K	176661	207		*	DISPVCOL	ADM A = ZERO WHEN ENTERED HERE
827,K	176662	340				LEA SO E = STRING VALUE
828,K	176663	015				INCP HL
829,K	176664	007				RET
830,K				*		
831,K	176665	106 332 373			DISPENDL	CALL DISPNEWL
832,K	176670	062 060			DISPEXIT	POP BC
833,K	176672	111 124 240				DPL BC,SEDOPTS
834,K	176675	250				XRA
835,K	176676	107 241				PS A,SEDOPTS+1
836,K	176700	022 060				POP XA
837,K	176702	007				RET

RESTORE THE XA REGISTER PAIR
FINALLY, EXIT THE DISPLAY ROUTINE

```

838,K
839,K 176703
840,K
841,K
842,K
843,K
844,K
845,K
846,K
847,K
848,K
849,K 176703 066 141 056 372
850,K 176707 016 040
851,K 176711 106 341 374
852,K 176714 113 024 012
853,K 176717 106 211 373
854,K 176722 106 140 373
855,K 176725 150 322 375
856,K
857,K 176730 310
858,K 176731 106 171 373
859,K 176734 076 357
860,K 176736 106 207 374
861,K 176741 301
862,K 176742 007
179,
180,

```

*
KEYCHAR

. GET A KEYBOARD CHARACTER

. ENTRY: DE = CURSOR POSITION TO BE FLASHED WHILE WAITING

. EXITS: A AND B = KEYBOARD CHARACTER

. C,H,L = SCRATCHED

. 5 STACK LEVELS USED (MAX) * LESS? *

```

HL      DSPBLNK
LB      SOW
CALL    DISPLAY
SUD     DSPBLN
CALL    DOSF66
KEYCHARW CALL  DOSF62
JTZ     KEYCHARW

```

BLANK OUT SOME SPACE FOR THE KEYIN
WITHOUT USING DOSF69 (BECAUSE OF CURSESRS)
CORRECT FOR UPDATING OF THE D-REGISTER
TURN ON THE CURSOR
WAIT FOR A CHARACTER TO BE INPUT

. LBA ECHO THE CHARACTER

```

CALL    DOSF63
LX      SECFLAGS>8
CALL    DOSF611N
LAB
RET

```

TURN THE CURSOR OFF (OVERWRITTEN BY 63)
EXIT WITH THE CHARACTER IN A AND B

. INC 1800MOVI

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- MOVI MACRO-ROM MEMORY TEST -

```

3,L
4,L
5,L
6,L 000004
7,L 040000
8,L
9,L
10,L 176743 022 040 061 062 070
11,L 176752 040 066 064 003
12,L 176756 060 064
13,L
14,L 176760
15,L 176760 106 112 370
16,L 176763 106 045 373
17,L 176766 106 004 374
18,L 176771 020
19,L 176772 036 007
20,L 176774 056 000
21,L 176776 365
22,L 176777 030
23,L 177000 046 000 036 100
24,L 177004 104 114 377
25,L
26,L 177007
27,L
28,L
29,L
30,L 177007 174 070
31,L 177011 046 006 036 111
32,L 177015 066 343 056 375
33,L 177021 106 337 374
34,L 177024 036 114
35,L 177026 106 312 374
36,L 177031 070
37,L
38,L 177032 066 264 056 363
39,L 177036 026 002
40,L 177040 077
41,L
42,L 177041 066 001 056 357
43,L 177045 046 215 036 377
44,L 177051 027
45,L
46,L 177052 076 000
47,L 177054 046 264
48,L 177056 250
49,L
50,L 177057 066 377 056 020
51,L 177063 376
52,L 177064 107 377
53,L 177066 277
54,L 177067 110 110 376

```

```

* 1800 MOVING INVERSIONS MEMORY TEST
*
MAXSTLS EQU 4 16K BLOCKS TESTED PER PASS
MAXSZ EQU MAXSTLS<12 MEMORY SIZE BEING TESTED (POWER OF 2)
MUST ALLOW 8K SPACE WITH MIN. MEMORY SIZE

*
MTMSG128 DC SEEDL,' 128K',SES
MTMSG64 DC ' 64',SES
MTINC DC '041' STEPS IN 4K CHUNCKS

*
TSTMEN
CALL C12345 SEE IF ENTERED MEMORY TEST LEGALLY
CALL SETUP
CALL DSPINIT INITIALIZE THE WORLD
BETA
LD 8-1 DISPLAY ERROR ADDRESS POINTER (HORIZ)
LH 0
LLH INITIALIZE THE PASS NUMBER
ALPHA
DE MAXSZ START INITIAL STEPSIZE AS 1
JMP MSTART FINISH INIT AS IF ENDED A PASS

*
MTSIZE
* SIZE CALCULATION DONE PER PASS
*
PUSH DE SAVE THE INCREMENT SIZE
DE 73<8+BL=5 POINT TO MEMORY LOCATION FOR 'K' COUNTER
HL MTMSG128
CALL DISPODS DISPLAY THE INITIAL MESSAGE
LD 76
CALL DSPCBL GET ADDRESS OF THE COUNTER IN DISPLAY RAM
PUSH

*
HL WRAPST SPECIAL WRAP AROUND MEMORY TEST
LC 2
STL

*
HL SVMEMP+1
DE MTSIZMPF SET INTERRUPT VECTOR FOR SIZE CALCULATION
DS DE,HL

*
LX 0>8 POINTER NEEDED LATER
LE SMST+PISTL ASSUME MEMORY OVER 64K BOUNDARY
XRA

*
HL 010377 SET 72K PHYS. TO ALL ONES
LML (OR 8K PHYS. IF WRAPPED)
PS A,000377 SET 8K PHYS. TO ZEROS
CPM READ 72K PHYS. (OR 8K) TO SEE IF NON-ZERO
JFZ MTSIZ32S NO WRAP AROUND, USING 32K CARDS

```

JAN 30, 1978 14:43 HJS

55,L									
56,L	177072	310				LBA		SET OPTIONS - DO BLANKS & WAIT ON DISPLAY	
57,L	177073	046 006 036 112				DE	74<8+BL-5	WRAPPED AROUND, USING 12K CARDS	
58,L	177077	066 352 056 375				HL	MTMSG64		
59,L	177103	106 341 374				CALL	DISPLAY	SO CHANGE THE MESSAGE	
60,L	177106	046 244				LE	SMST+SMSTL+UMSTL	AND START MEMORY SIZE TEST FROM THERE	
61,L									
62,L	177110	364				MTSIZ32S	LLE	MOVE SIZE START TO L	
63,L									
64,L	177111					MTSIZLP			
65,L	177111	174 060				POP	DE	GET MEMORY POINTER	
66,L	177113	174 070				PUSH	DE	AND KEEP IT THERE	
67,L	177115	070				PUSH	HL	SAVE THE TABLE POINTER	
68,L	177116	066 357 056 375				HL	MTINC+1		
69,L	177122	250				XRA		(CLEAR CARRY)	
70,L	177123	026 002				LC	2		
71,L	177125	016 060				LB	'0'	UPDATE THE 'K' COUNTER IN FOUR'S	
72,L	177127	062 041				DFSB			
73,L	177131	100 140 376				JFC	MTSIZLP0	WILL NOT GET CARRY UNLESS CROSS 100K	
74,L	177134	006 040				LA	' '		
75,L	177136	174 370				LMA	DE	DONE FOR COSMETIC REASONS ONLY	
76,L									
77,L	177140	060				MTSIZLP0	POP		
78,L	177141	056 363				LH	SMST>8	(SAFETY, DON'T TRUST MEMORY)	
79,L	177143	035				DECP	HL	TRY THE NEXT STL ENTRY	
80,L	177144	176 074 233				CPL	SMST+6	MINIMAL SIZE AT 24K	
81,L	177147	140 173 376				JTC	MTSIZIT		
82,L									
83,L	177152	026 001				LC	1		
84,L	177154	077				STL			
85,L									
86,L	177155	016 377				LB	0377	LOAD NON-ZERO IN SOME MEMORY	
87,L	177157	321				LCB			
88,L	177160	111 126 000				DPS	BC,0	TO SEE IF IT IS THERE	
89,L	177163	111 124 000				DPL	BC,0	(X-REG IS ZERO FROM ABOVE)	
90,L	177166	062 261				ORBC		IF ZERO, NOT FOUND ANY YET	
91,L	177170	150 111 376				JTZ	MTSIZLP		
92,L									
93,L	177173	015				MTSIZIT	INCP	HL	CORRECT POINTER TO FIRST MISSING SECTOR
94,L	177174	306				LAL			
95,L	177175	046 201 036 377				DE	MERX		
96,L	177201	066 001 056 357				HL	SVMEMP+1	SET THE VECTOR FOR THE MEMORY TEST	
97,L	177205	027				DS	DE,HL		
98,L	177206	060				POP		DON'T NEED THE DISPLAY POINTER	
99,L	177207	174 060				POP	DE	RESTORE INCREMENT SIZE	

100	L				
101	L	177211			
102	L				
103	L				
104	L				
105	L				
106	L				
107	L				
108	L	177211	024	004	
109	L				
110	L	177213	074	227	
111	L	177215	100	275	376
112	L				
113	L	177220	074	226	
114	L	177222	006	227	
115	L	177224	150	275	376
116	L				
117	L				
118	L	177227	174	070	
119	L	177231	066	231	056 363
120	L	177235	026	002	
121	L	177237	077		
122	L	177240	250		
123	L	177241	310		
124	L	177242	320		
125	L	177243	330		
126	L	177244	340		
127	L	177245	056	320	
128	L	177247	360		
129	L	177250	076	040	
130	L				
131	L	177252	021		
132	L	177253	117	024	001
133	L	177256	110	252	376
134	L				
135	L	177261	174	060	
136	L	177263	066	231	056 363
137	L	177267	026	322	
138	L	177271	062	077	
139	L	177273	006	225	
140	L				
141	L	177275	076	000	
142	L	177277	117	200	
143	L	177301	360		
144	L	177302	056	363	
145	L	177304	026	004	
146	L	177306	077		

```

+
MTMAINLP
.
. MAIN LOOP OF THE MEMORY TEST EXECUTED PER MEMORY BLOCK
.
. HERE, A = ADDRESS+1 OF LAST ENTRY IN STL TABLE THAT SHOWS MEMORY
.
.
MTMNBK SU MAXSTLS BACK UP STL ENTRIES FOR NEXT BLOCK OF TABL
. IF USED MAXSTLS=1 WILL GENERATE AN OVERLAP
CP UMST REACHED THE BEGINNING OF THE NORMAL BLOCK?
JFC MTMNXT NO, JUST ANOTHER SECTION
.
CP UMST-1 BACKED UP INTO SPECIAL AREA?
LA UMST IF BACKUP WOULD CAUSE SKIP OF 12-15K PHYS
JTZ MTMNXT DO A SPECIAL PASS TO GET IT
.
.
PUSH DE REACHED SPECIAL LAST STL SECTION
HL SMST+MAXSTLS SAVE ADDRESS INCREMENT SIZE
LC SMSTL POINT TO FIRST UNUSED SECTOR
STL THAT AVAILABLE IN SPECIAL PASS
XRA
LBA
LCA
LDA
LEA
LH 0150000>8
LLA
LX 0170000-0150000>8 COUNT BLOCKS OF 256
.
MTMSP0 BT TO BE MOVED TO SAFE PLACE
SUX 1
JFZ MTMSP0
.
POP DE RESTORE DE FROM OLD AREA (SAFER?)
HL SMST+MAXSTLS MOVE SECTOR TABLES FOR SPECIAL AREA
LC (SEDSPBF>8.AND.0360)+SMSTL
STLOC THAT USED IN SPECIAL PASS
LA SMST
.
MTMNXT LX 0 MOVE THE STL BLOCK # TO THE X-REGISTER
ADAX
LLA
LH SMST>8 POINT TO THE STL TABLE
LC MAXSTLS
STL WILL NOW WORK ON THOSE SECTORS

```

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- MOVI MACRO-ROM MEMORY TEST -

147,L
148,L 177307 106 102 373
149,L 177312 160 366 377
150,L
151,L 177315 250
152,L 177316 310
153,L 177317 320
154,L 177320 066 376 056 077
155,L 177324 111 027
156,L 177326 117 035
157,L 177330 100 324 376
158,L
159,L 177333 026 240 016 377
160,L 177337 066 001 056 357
161,L 177343 111 027
162,L 177345 310
163,L
164,L 177346 026 001
165,L 177350 360
166,L 177351 350
167,L
168,L
169,L
170,L
171,L
172,L

*
CALL DOSF60N SHOULD THE MEMORY TEST BE ENDED?
JTS MTEND YES, JUST END IT HERE
.
XRA ** START MEMORY TEST PHASE **
LBA
LCA
HL MAXSZ=2
MTMNX0 DS BC,HL INITIALLY, THE MEMORY MUST BE ZERO
DECP HL,2
JFC MTMNX0
.
BC MERPRF
HL SVMEMP+1 SET AN ADDRESS FOR PARITY ERRORS IN MEMORY
DS RC,HL
LBA
.
MFNXZ LC 1
MFNXB LLA
LHA
.
NOW: HL, B, AND A ARE ZERO
C IS INITIALIZED TO FIRST BIT TO TEST
X IS STL TABLE POINTER
DE IS INCREMENT
.


```

173,L
174,L 177352
175,L
176,L
177,L
178,L
179,L
180,L
181,L
182,L
183,L
184,L
185,L
186,L
187,L
188,L
189,L
190,L
191,L
192,L
193,L
194,L 177352 307
195,L 177353 271
196,L 177354 112 237 377
197,L 177357 252
198,L 177360 370
199,L 177361 277
200,L 177362 112 237 377
201,L
202,L 177365 176 204
203,L 177367 115 213
204,L 177371 115 074 100
205,L 177374 140 352 376
206,L 177377 015
207,L 177400 115 044 077
208,L 177403 115 273
209,L 177405 110 352 376
210,L 177410 174 276
211,L 177412 110 352 376
212,L
213,L 177415 250
214,L 177416 111 252
215,L 177420 062 202
216,L 177422 110 350 376
217,L 177425 153
218,L 177426 111 261
219,L 177430 160 346 376
220,L

```

```

+
MFWDLP
.
. ASCENDING THROUGH THE MEMORY PART OF THE MEMORY TEST
. NOTE:
.   FORWARD ADDRESS SEQUENCE IS:
.   0, 1, 2, 3, 4, ... -2, -1. (ACTUALLY IN 'DE' STEPS)
. NOTE:
.   IT ALWAYS STARTS AT ZERO AND ENDS WHEN REACHS 'DE' WITH OVERFLOW
.
. INTERNALLY:
. ALPHA  A      WORK REGISTER
.         B      EXPECTED DATA
.         C      BIT UNDER TEST
.         DE     ADDRESS INCREMENT
.         HL     CURRENT ADDRESS BEING TESTED
.         X      STL SEGMENT BEING TESTED
.
. BETA   HL      PASS COUNTER
.         D      ADDRESS ERROR DISPLAY POINTER (HORIZ POSN)
.
.
. LAM          GET THE DATA (CAN CREATE PARITY FAULT)
. CPB
. CFZ          MERONG      WRONG, DATA CHANGED
. XRC          SET BIT BEING TESTED
. LMA
. CPM
. CFZ          MERONG      STILL SET? (PARITY FAULT POSSIBLE)
.                               NO, DATA CHANGED RATHER QUICKLY
.
MFWDNXT
. ADEL        ERRORS CONTINUE FROM HERE
. ACDH        UPDATE HL USING DE
. CPH         MAXSZ>8      WRAP AROUND?
. JTC         MFWDLP      NO, GO BACK FOR MORE
. INCP        HL          CARRY WRAPAROUND FROM MSB
. NDH         MAXSZ-1>8    AND KEEP THE ADDRESS IN RANGE
. CPDH
. JFZ         MFWDLP      MSB DOESN'T MATCH STEP SIZE, NOT THE END
. CPLE
. JFZ         MFWDLP      LSB DOESN'T MATCH STEP SIZE, NOT THE END
.
. XRA         RESET THE ADDRESS TO ZERO
. XRCB        SET THE BIT FOR THE NEXT SUB-PASS
. ADCC        SHIFT FOR NEXT BIT (SAFER THAN SHIFT)
. JFZ         MFNXB      CONTINUE TILL ALL BITS
. EX          CLICK
. ORBR
. JTS         MFNXZ      GO FROM 0 TO 1 AND BACK TO ZERO
.                               (SAFER THAN JFZ)

```

```

221,L
222,L 177433
223,L
224,L
225,L
226,L
227,L
228,L 177433 026 201 016 377
229,L 177437 066 001 056 357
230,L 177443 111 027
231,L
232,L 177445 022 070
233,L 177447 062 060
234,L 177451 301
235,L 177452 074 227
236,L 177454 100 211 376
237,L
238,L 177457 174 070
239,L 177461 046 224 036 377
240,L 177465 027
241,L
242,L 177466 250
243,L 177467 310
244,L 177470 320
245,L 177471 330
246,L 177472 340
247,L 177473 056 320
248,L 177475 360
249,L 177476 076 040
250,L 177500 021
251,L 177501 117 024 001
252,L 177504 110 100 377
253,L 177507 106 003 373
254,L 177512 174 060
255,L
256,L 177514
257,L 177514 026 201 016 377
258,L 177520 066 001 056 357
259,L 177524 111 027
260,L
261,L 177526 020
262,L 177527 015
263,L 177530 315
264,L 177531 326
265,L 177532 066 006
266,L 177534 046 005 036 117
267,L 177540 106 100 372
268,L 177543 111 020
269,L 177545 015
270,L 177546 030
271,L
272,L 177547 303

```

+
MTMDON

. A MAJOR PASS HAS COMPLETED.
. CONTINUE TILL ALL MEMORY BLOCKS DONE
. THEN INCREMENT PASS COUNTER, SET NEW ADDRESS INCREMENT AND DO NEXT

```

.      BC      MERY
      HL      SVMEMP+1      ANY MEMORY ERROR HERE IS FATAL
      DS      BC,HL

.      PUSH    XA      GET THE STL TABLE POINTER
      POP     BC
      LAB     FROM THE X-REGISTER
      CP      UMST      (JFC SAFER THAN JFZ)
      JFC     MTMNBK     CONTINUE TILL ALL BLOCKS DONE

.      PUSH    DE      SAVE ADDRESS INCREMENT
      DE      MTDONMPF   IGNORE MEMORY FAULTS THAT HAPPEN HERE
      DS      DE,HL

```

```

.      XRA     MOVE THE DISPLAY BUFFER, STACK SAVE AREA
      LBA
      LCA     INTERRUPT VECTOR AND OTHER NECESSARY INFO
      LDA
      LEA     BACK TO THE PHYSICAL MEMORY
      LH      0150000>8
      LLA
      LX      0170000-0150000>8
      BT      MTMD0
      SUX     1      COUNT THE BLOCKS GOING BY
      JFZ     MTMD0
      CALL    MTSETUP  RESTORE SECTOR TABLE & INTERRUPT VECTOR
      POP     DE

```

. MSTART

```

      BC      MERY
      HL      SVMEMP+1      ANY MEMORY ERROR HERE IS FATAL
      DS      BC,HL

.      BETA
      INCP    HL      UPDATE PASS COUNTER
      LBH
      LCL
      LL      6      DISPLAY THE NEW PASS COUNTER VALUE
      DE      79<8+BL-6  (DESTROYS MOST REGISTERS)
      CALL    DSPOCT
      BETAL   RE-LOAD BETA REGS, RESTORES WHAT DESTROYED
      INCP    HL
      ALPHA

.      LAD

```

273,L 177550 174 204
274,L 177552 210
275,L 177553 004 300
276,L 177555 174 014 000
277,L 177560 044 077
278,L 177562 330
279,L 177563 350
280,L 177564 364
281,L 177565 035
282,L 177566 140 201 377
283,L 177571 176 244
284,L 177573 115 243
285,L 177575 035
286,L 177576 140 007 376
287,L

ADEF
ACA
AD
ACE
ND
LDA
LHA
LLE
DECP
JTC
NDEL
NDDH
DECP
JTC

GET NEXT ASCENDING STEP SIZE
MAXSZ-1,XOR,(-1)>8 GENERATE CORRECT NUMBER FOR CARRY AROUND
0
MAXSZ-1>8

HL
MERX

DOUBLE CHECK THAT ONLY ONE BIT SET
HL SHOULD BECOME (POWER OF 2) - 1
CATCH DE BECOMING ZERO AS A FATAL ERROR

HL
MTSIZE

SHOULD RESULT IN ZERO IF WAS POWER OF 2
SHOULD SET CARRY IF WAS ZERO
OK, RE-CALCULATE MEMORY SIZE FOR NEXT PASS
** ERROR, WAS NOT POWER OF 2 **

```

288,L
289,L
290,L
291,L 177601 151
292,L
293,L 177602 106 003 373
294,L 177605 106 242 372
295,L 177610 066 252
296,L 177612 104 055 362
297,L
298,L 177615 174 060
299,L 177617 174 060
300,L 177621 104 173 376
301,L
302,L 177624 302
303,L 177625 062 060
304,L 177627 062 060
305,L 177631 320
306,L 177632 250
307,L 177633 310
308,L 177634 104 100 377
309,L
310,L
311,L
312,L 177637 070
313,L
314,L 177640 070
315,L 177641 022 070
316,L 177643 020
317,L 177644 062 060
318,L 177646 301
319,L 177647 062 060
320,L 177651 070
321,L 177652 056 363
322,L 177654 360
323,L
324,L 177655 301
325,L 177656 044 360
326,L 177660 111 250
327,L 177662 012 012 012 012
328,L 177666 074 004
329,L 177670 100 201 377
330,L
331,L 177673 017
332,L 177674 307
333,L 177675 044 360
334,L 177677 111 260
335,L 177701 307
336,L 177702 002
337,L 177703 044 003
338,L 177705 004 060
339,L 177707 022 070

```

```

+
. ERROR PROCESSING ROUTINES
.
MERX      EX      BEEP      EXCEPTIONAL MEMORY ERROR THAT DESTROYED
.                                OPERATION OF THE MEMORY TEST HAPPENED
.                                CALL      MTSETUP
.                                CALL      SSTATE      RESTORE STATE TO GOOD VALUES (I HOPE)
.                                LL        MERXMSG      POINT TO MESSAGE (WITH OTHER MESSAGES)
.                                JMP        ERROR
.
*
MTSIZMPF  POP      DE      SKIP P-COUNTER
.                                POP      DE      AND ERROR LOCATION
.                                JMP      MTSIZIT     BUT KEEP TRYING
.
*
MTDONMPF  LAC      INTO A SAFE PLACE
.                                POP      BC      GET WASTED INFO OUT OF THE WAY
.                                POP      BC
.                                LCA
.                                XRA
.                                LBA
.                                JMP      MTMD0      AND TRY TO RECOVER
.
. ERROR IN DATA OR A PARITY FAULT WHILE ACTUALLY DOING MEMORY TEST OPERATIONS
.
MERONG     PUSH      WRONG DATA, CORRECT STACK TO MATCH PARITY
.
MERPRF     PUSH      HL      COPY REAL ADDRESS OVER TO BETA REGS
.                                PUSH     XA      AND THE STL TABLE POINTER
.                                BETA
.                                POP      BC
.                                LAB      NOW HAVE TABLE POINTER (FROM ALPHA = X)
.                                POP      BC      AND ADDRESS TO CALCULATE PHYSICAL ADDRESS
.                                PUSH     HL      SAVE PASS COUNTER
.                                LH       SMST>8
.                                LLA      GET STL TABLE POINTER
.
.                                LAB
.                                ND       0360      GET STL INDEX BITS
.                                XRAB     CLEAR THEM OUT OF THE ADDRESS
.                                SRN      4
.                                CP       MAXSTLS
.                                JFC      MERX      GOT ERROR OUTSIDE OF MEMORY BEING TESTED?
.
.                                INCP     HL,A
.                                LAM
.                                ND       0360      GET STL ENTRY THAT IN ERROR AREA
.                                ORAB
.                                LAM      GENERATED PHYSICAL ADDRESS (LS15 BITS)
.                                SLC
.                                ND       3
.                                AD       '0'      GET BITS 16 & 17 OF PHYSICAL ADDRESS
.                                PUSH     XA

```

340,L	177711	046 013	LE	BL	
341,L	177713	066 005	LL	5	OUTPUT 5 LS DIGITS OF THE PHYSICAL ADDRESS
342,L	177715	106 160 372	CALL	DSPOCT	
343,L	177720	022 060	POP	XA	
344,L	177722	370	LMA		AND THE MS DIGIT ALSO
345,L					
346,L	177723	113 004 015	ADD	5+8	5 FOR DSPOCT & 8 FOR NEXT POSN
347,L	177726	113 074 107	CPD	80-8-1	GO TO NEXT DISPLAY POSITION
348,L	177731	140 347 377	JTC	MERR1	
349,L					
350,L	177734	106 102 373	MERR0 CALL	DOSF60N	WAIT FOR THE DISPLAY KEY TO BE UP
351,L	177737	170 334 377	JTP	MERR0	
352,L	177742	106 345 373	CALL	DOSF67	ROLL UP
353,L	177745	036 007	LD	8-1	POSITION FOR NEXT ERROR ADDRESS
354,L					
355,L	177747	060	MERR1 POP	HL	RESTORE PASS COUNTER
356,L	177750	062 060	POP	BC	IGNORE P-COUNTER
357,L	177752	062 060	POP	RC	DISCARD LOCATION OF FAULT
358,L	177754	030	ALPHA		GO BACK TO NORMAL REGISTERS
359,L	177755	115 044 077	NDH	MAXSZ-1>8	AFTER BEING IN MEMORY, MAKE IT IN RANGE
360,L					AS A MINIMAL ASSURANCE OF ACCURACY
361,L	177760	301	LAB		
362,L	177761	252	XRC		
363,L	177762	370	LMA		
364,L	177763	104 365 376	JMP	MFWDNXT	SET MEMORY LOCATION TO WHAT IT SHOULD BE
365,L					AND RETURN FROM THE ERROR
366,L	177766	106 003 373	MTEND CALL	MTSETUP	RESTORE THE SECTOR TABLE TO NORMAL STATE
367,L	177771	104 127 367	JMP	DEBUG	
181.					

MACRO-PROCESSOR SYSTEM MACRO ROM CODE - HSP/HJS - 78JUL20 11:44
- END OF 1800 MACRO ROM -

184,
185,
186,
187,
188,
189,
190,
191,
192,
193,
194,
195, 177774 000
196, 177775 000 000
197, 177777 011
198,
199, 170000

*
IFNE \$>8,0177777>8 (CHECK IF LOADED TO LAST PAGE)
ERR MACRO ROM NOT CORRECT SIZE, TOO BIG OR TOO SMALL
XIF
*
; PAD TO THE END OF THE PAGE WITH ZEROS AND PUT THE VERSION OF THE MACRO
; ROM IN THE LAST BYTE IN THE PAGE AND
; THE ROM TYPE IN THE SECOND LAST BYTE
*
LIST =G
RPT =S-4,AND,255
DC 000
DC ROMTYPE
DA *ROMCRC
\$MACVRS* DC \$MACROM
*
END MACROM

[illegible]

177777	\$MACVRS	*197											
000215	\$NL	*653:K	787:K										
000203	\$NS	*649:K	220:K	221:K	223:K	224:K	225:K	227:K	229:K	231:K	233:K	234:K	239:K
		709:K	788:K										
000233	\$O	*658:K	711:K	792:K									
000200	\$OI	*660:K	450:K										
170176	\$ONLINE	*109:K											
000100	\$OS	*661:K	684:K										
000040	\$OW	*662:K	684:K	725:K	850:K								
000024	\$RD	*646:K	794:K										
170170	\$READ	*107:K											
170201	\$RESTORE	*110:K											
173246	\$RESTORL	*342:K	348:K										
173233	\$RESTORX	*333:K	24:K										
000023	\$RU	*645:K	793:K										
156605	\$SAVBC	*109	69:K	96:K	150:K	158:K	283:K	308:K					
156603	\$SAVXA	*108	68:K	97:K	281:K	282:K	309:K	310:K					
156620	\$SECTOR	*116	305:K	422:K	431:K								
170162	\$SEEK	*105:K											
172637	\$SEEKLOP	*45:K	36:K	49:K									
172620	\$SEEKOK	*30:K	22:K										
172633	\$SEEKOUT	*40:K	336:K										
170215	\$STATUS	*115:K											
170165	\$STEP	*106:K											
172667	\$STEPWTL	*77:K	90:K										
172722	\$STEPXIT	*96:K	86:K										
156621	\$TRACK	*117	136:K	20:K	34:K	334:K							
000013	\$V	*641:K	784:K										
000213	\$VA	*652:K	38:K	786:K									
170173	\$WRITE	*108:K											
000000	\$AL	*23:K											
000000	\$AN	*22:K											
000140	\$ARFC	*30:K											
000150	\$ARFE	*32:K											
000170	\$ARFP	*34:K											
000160	\$ARFS	*33:K											
000150	\$ARFZ	*31:K											
000100	\$ARTC	*25:K											
000110	\$ARTE	*27:K											
000130	\$ARTP	*29:K											
000120	\$ARTS	*28:K											
000110	\$ARTZ	*26:K											
174610	\$ADRDE1	*669:K	667:K										
174600	\$ADRDEV	*659:K	166:K	176:K	177:K								
000000	\$ASCII	*23	52:K	114:K	197:K	200:K	342:K						
170561	\$AVIOLM	*230:K	313:K										
171020	\$AVIOLT	*312:K	179:K	180:K									
000001	\$B0	*6:K	17:K	37:K	59:K	72:K	88:K	100:K	111:K	86:K	97:K	111:K	127:K
		167:K											
000002	\$B1	*7:K	18:K	38:K	60:K	73:K	101:K	112:K	87:K	98:K	112:K	128:K	168:K
000004	\$B2	*8:K	39:K	61:K	74:K	102:K	113:K	88:K	99:K	113:K	129:K	169:K	
000010	\$B3	*9:K	21:K	40:K	51:K	75:K	91:K	103:K	114:K	89:K	100:K	114:K	130:K

[illegible]

000061	CODMIN	*132:C												
000071	CODMOUT	*131:C												
000127	CODWRT	*133:C												
153600	CURADR	*62	342:F	135:H	137:H	36:J	63:J	81:J	112:J	395:J	400:J	408:J	410:J	
		424:J	431:J	436:J	451:J	475:J	567:J	575:J	621:J	641:J	650:J	680:J	783:J	
		788:J												
000010	CURINBUF	*91:C	276:K	290:K	572:K	579:K								
000001	CURONOFF	*88:C	237:K	271:K	514:K									
153602	CUROUT	*63	723:J	727:J										
176237	CURSES	*553:K	83:F	253:K	688:K									
176225	CURSESRS	*540:K	335:K	399:K	430:K	477:K	511:K							
000000	CUSOR	*122:C	405:F											
153603	CURSTA	*64	679:J											
170110	D\$BLINK	*85:F												
170105	D\$BLNKDE	*84:F												
170077	D\$CBL	*82:F												
170102	D\$CURSES	*83:F												
173527	DEBUG	*40:J	617:F	789:J	367:L									
173533	DEBUGI	*48:J	647:F											
174257	DECAOR	*427:J	173:J											
174271	DECADV	*434:J	432:J											
000177	DEL	*127:C												
176647	DISPBEEP	*811:K	797:K											
176651	DISPCLIK	*814:K	798:K											
176352	DISPCPRP	*691:K	777:K											
176353	DISPCPSL	*692:K	735:K	774:K										
176447	DISPDCC	*734:K	726:K											
176337	DISPDOO	*684:K	80:F	131:H	161:H	33:L								
176435	DISPOSPW	*728:K	732:K											
176665	DISPENDL	*831:K	799:K											
176670	DISPEXIT	*832:K	800:K											
176413	DISPFCLP	*715:K	722:K											
176462	DISPFORC	*743:K	706:K											
176554	DISPFTAB	*779:K	714:K											
176407	DISPFIINC	*714:K	708:K											
176640	DISPGNSA	*803:K	788:K											
176642	DISPGOPT	*806:K	792:K											
176653	DISPHADJ	*817:K	785:K											
176654	DISPHCOL	*819:K	783:K											
176357	DISPLANC	*695:K	772:K											
176341	DISPLAY	*686:K	79:F	341:F	820:J	851:K	59:L							
175732	DISPNEWL	*309:K	787:K	831:K										
176546	DISPOFFS	*776:K	755:K											
176465	DISPQUT	*748:K	710:K	712:K	718:K									
176525	DISPSKIP	*767:K	764:K											
176660	DISPVADJ	*824:K												

170127	DOSFNC64	*91IF																		
170132	DOSFNC65	*92IF																		
170135	DOSFNC66	*93IF																		
170140	DOSFNC67	*94IF																		
170143	DOSFNC68	*95IF																		
170146	DOSFNC69	*96IF																		
170151	DOSFNC6A	*97IF																		
170154	DOSFNC6B	*98IF																		
170157	DOSFNC6C	*99IF																		
173726	DOCMOL	*147IJ	142IJ																	
154000	DOSBUF	*37IB	97																	
156000	DOSEXT	*35IB	36IB	99	103															
175477	DOSF60	*79IK	87IF																	
175502	DOSF60N	*93IK	53IG	73IH	144IH	148IL	350IL													
175525	DOSF61	*115IK	88IF	811IJ																
176144	DOSF610	*465IK	97IF	794IK																
176204	DOSF611	*498IK	98IF																	
176207	DOSF611N	*512IK	860IK																	
176216	DOSF612	*518IK	99IF																	
175540	DOSF62	*132IK	89IF	854IK																
175543	DOSF62N	*151IK	74IF																	
175571	DOSF63	*165IK	90IF	858IK																
175577	DOSF64	*195IK	91IF	378IK	789IK															
175604	DOSF65	*209IK	92IF	790IK																
175611	DOSF66	*223IK	93IF	853IK																
175745	DOSF67	*317IK	94IF	793IK	352IL															
176046	DOSF68	*381IK	95IF	796IK																
176053	DOSF68LP	*402IK	809IJ	414IK																
176105	DOSF69	*418IK	96IF	795IK																
000006	DSKWRT	*179II	199II																	
000007	DSKWV	*178II																		
000012	DSPBLN	*854IJ	852IK																	
175141	DSPBLNK	*849IJ	854IJ	849IK																
173533	DSPCAD	*59IJ	38IJ	100IJ																
176312	DSPCBL	*598IK	82IF	90IJ	876IJ	189IK	284IK	336IK	402IK	431IK	478IK	583IK	754IK							
		35IL																		
176004	DSPINIT	*356IK	78IF																	

172354	FALIGN	*62IH					
172363	FALoop	*70IH	74IH				
172431	FAUTO	*98IH	52IH	127IH			
172433	FAUTO0	*104IH	125IH				
172435	FAUTO1	*110IH	117IH	120IH			
172476	FAUTOER	*129IH	112IH				
172572	FAUTOMSG	*167IH	129IH				
172525	FCHECKR	*140IH	84IH	93IH	95IH	111IH	145IH
000040	FCINDP	*66ID					
000020	FCINDT	*65ID					
000000	FCINST	*64ID					
000060	FCLEAR	*67ID	384II				
000260	FCOINT	*75ID					
000220	FCOMOD	*73ID	74II	127II	366II		
000160	FCOSYL	*71ID					
000140	FCOSYM	*70ID					
000100	FCOTDP	*68ID					
000120	FCOTUP	*69ID					
000240	FCOUTC	*74ID					
000300	FCRHDR	*76ID					
000320	FCRTIM	*77ID					
000200	FCWDCG	*72ID					
000360	FCWDGP	*79ID					
000340	FCWRTN	*78ID					
000010	FDDBL	*192ID					
172205	FDIAG	*6IH	184IJ				
172213	FDIAGDR	*36IH	42IH	46IH	59IH		
172512	FDIAGEX	*134IH	146IH	149IH			
000010	FDIND	*183ID	224II				
000000	FDINS	*182ID					
000004	FDLTD	*180ID					
000070	FDMASK	*195ID					
000040	FDNINV	*194ID					
000020	FDNOT	*193ID					
000050	FDOTD	*187ID	189II				
000040	FDOTS	*186ID					
000070	FDOUT	*189ID					
000003	FDSMSK	*191ID					
000030	FDVRD	*185ID	204II				
000020	FDVRS	*184ID					
000060	FDWPI	*188ID	185II				
000010	FFDBL	*204ID	194II	279II			
172401	FFIND	*78IH	56IH	85IH			
000070	FFMASK	*209ID					
000020	FFRDLTD	*200ID					
000000	FFREAD	*199ID	70IH	279II			
000040	FFRMSK	*210ID					
000000	FFSGL	*203ID					
000010	FFSMSK	*212ID					
000070	FFSYNC	*207ID					
000060	FFWDCG	*206ID					
000020	FFWMSK	*211ID					

000040	FFWRITE	*201:0	194:I	
172557	FGETRSP	*160:0H	37:0H	48:0H
000017	FIADR	*135:0D		
000060	FIINDX	*138:0D		
000160	FINUM	*136:0D		
000160	FIPNTR	*142:0D		
000100	FISTEP	*139:0D		
000140	FITRER	*141:0D		
000120	FITROK	*140:0D		
000001	FKINDX	*111:0D	120:0D	
000100	FKLOFF	*117:0D	385:0I	
000200	FKLON	*118:0D		
000077	FKMAST	*120:0D	385:0I	
000020	FKPNTR	*115:0D	120:0D	
000040	FKRWMF	*116:0D	120:0D	
000002	FKSTEP	*112:0D	120:0D	
000010	FKTRER	*114:0D	120:0D	
000004	FKTROK	*113:0D	120:0D	
000001	FMINDX	*127:0D		
000020	FMPNTR	*131:0D		
000002	FMSTEP	*128:0D		
000010	FMTRER	*130:0D		
000004	FMTROK	*129:0D		
000200	F043GT	*104:0D	106:0D	107:0D
000100	F043LE	*103:0D	107:0D	
177524	F043MGC	*106:0D		
000300	F043MSK	*107:0D		
000001	F0DR0	*97:0D	622:0F	434:0I
000002	F0DR1	*98:0D	624:0F	435:0I
000004	F0LOAD	*99:0D		
000020	F0MVIN	*101:0D	43:0I	
000040	F0MVOT	*102:0D	35:0I	342:0I
000010	FOUNLD	*100:0D		
000400	FPLDTA	*154:0D	157:0D	
000016	FPLHDR	*150:0D		
000406	FPLRDTA	*157:0D	160:0D	
000424	FPLWDBL	*160:0D	163:0D	
000430	FPLWSGL	*163:0D		
000200	FPTDTA	*153:0D	156:0D	
000000	FPTHDR	*149:0D	151:0D	
000176	FPTRDTA	*156:0D	159:0D	
000002	FPTRKH	*151:0D		
000162	FPTWDBL	*159:0D	162:0D	
000156	FPTWSGL	*162:0D		
000002	FRBUSY	*168:0D	302:0I	
000001	FRDRV	*167:0D		
000004	FRINDX	*169:0D	306:0I	
000100	FRIXCT	*174:0D		
000200	FRIXCX	*176:0D		
172413	FSEEK	*87:0H	55:0H	
172414	FSEEK1	*92:0H	96:0H	
000010	FSFPRO	*89:0D	157:0I	

000020	FSGAP	*90:ID												
000200	FSNSYN	*93:ID												
000001	FSONLN	*86:ID	84:I	119:I	120:I	153:I	297:I							
000040	FSSTIP	*91:ID	89:I											
000002	FSTRIP	*87:ID	119:I	120:I										
000100	FSTRK0	*92:ID	93:I	94:I										
000004	FSWDIS	*88:ID												
173607	GETCMD	*102:J	125:J	316:J										
174125	GETCME	*304:J	201:J	210:J	211:J	251:J	255:J	257:J	258:J	263:J	271:J	277:J	278:J	
		280:J	281:J	282:J	283:J	327:J	329:J	562:J	653:J					
173616	GETCML	*109:J	309:J	319:J	339:J									
174150	GETDIG	*321:J	137:J	139:J	145:J									
174623	GETSTA	*678:J	672:J	721:J										
171644	HMST	*552:F	552:F	553:F	554:F	555:F	556:F	557:F	558:F	559:F	560:F	561:F	562:F	
		563:F	564:F	565:F	566:F	567:F								
175071	IDENT	*815:J	129:J											
175115	IDENT0	*830:J	825:J											
175130	IDENTM	*838:J	819:J											
175131	IDENTS	*843:J	850:J	852:J										
000017	IDMIC	*852:J	824:J											
000013	IDPROC	*850:J	829:J											
174211	INCAOR	*391:J	195:J											
174223	INCADV	*398:J	396:J											
170505	INPARM	*225:F	301:F											
170770	INPART	*300:F	173:F	174:F										
170606	IVIOLM	*233:F	317:F											
171030	IVIOLT	*316:F	181:F	182:F										
174773	JUMP	*776:J	196:J											
175033	JUMPIT	*794:J	792:J											
171165	KBDSPILP	*397:F	402:F											
171117	KBDSPINI	*367:F	75:F	265:F	612:F	665:F								
176703	KEYCHAR	*839:K	163:H	110:J										
176722	KEYCHARW	*854:K	855:K											
153606	KEYINF	*66												
000000	LC	*41	130:H	206:K	220:K	309:K	411:K							
174233	LINK	*403:J	203:J											
174241	LINKADR	*409:J	407:J											
172115	LOADIPL	*64:G	48:G											
170000	MACROM	*26:F	199											
000030	MAXLIN	*83:K	99:B	205:K	348:K	368:K	490:K	620:K						
000120	MAXPOS	*84:K	337:J	363:K	447:K	615:K	771:K							
000032	MAXSECT	*27:K	116:H	119:H										
000004	MAXSTLS	*6:L	7:L	108:L	119:L	136:L	145:L	328:L						
040000	MAXSZ	*7:L	23:L	154:L	204:L	207:L	275:L	277:L	359:L					
000114	MAXTRAK	*26:K	124:H	339:J										
170470	MEMPAM	*224:F	279:F	336:F										
170731	MEMPAT	*278:F	171:F	172:F										
177637	MERONG	*312:L	196:L	200:L										
177640	MERPRF	*314:L	159:L											
177734	MERR0	*350:L	351:L											
177747	MERR1	*355:L	348:L											
170641	MERROR	*237:F	220:F	221:F	227:F	231:F	239:F							

177601	MERX	*2911L	951L	2281L	2571L	2821L	3291L											
170652	MERXMSG	*2391F	2951L															
177350	MFNXB	*1651L	2161L															
177346	MFNXZ	*1641L	2191L															
177352	MFWDLP	*1741L	2051L	2091L	2111L													
177365	MFWDNXT	*2021L	3641L															
170625	MINSTE	*2361F	2331F	2341F														
174324	MODAGN	*4561J	1331J															
174227	MODIFNXT	*4001J	4691J															
174335	MODIFX	*4631J	4611J															
174343	MODIFY	*4711J	2041J	4541J														
174340	MODINC	*4651J	1311J															
153604	MODVAL	*65	4621J	4631J														
170533	MPARIT	*2271F	2231F	2241F	2251F													
170573	MPROTE	*2311F	2291F															
177514	MSTART	*2561L	241L															
170736	MSTPAT	*2811F	2761F															
177624	MTDONMPF	*3021L	2391L															
177766	MTEND	*3661L	1491L															
176756	MTINC	*121L	681L															
177211	MTMAINLP	*1011L																
177500	MTMD0	*2501L	2521L	3081L														
177433	MTMDON	*2221L																
177211	MTMNBK	*1081L	2361L															
177324	MTMNX0	*1551L	1571L															
177275	MTMNXT	*1411L	1111L	1151L														
176743	MTMSG128	*101L	321L															
176752	MTMSG64	*111L	581L															
177252	MTMSP0	*1311L	1331L															
167534	MTRFLG	*711B																
167535	MTRLEN	*721B																
167536	MTRPNT	*731B																
175403	MTSETUP	*9941J	10271J	2531L	2931L	3661L												
175405	MTSETX	*10001J	391F															
177110	MTSIZ32S	*621L	541L															
177007	MTSIZE	*261L	2861L															
177173	MTSIZIT	*931L	811L	3001L														
177111	MTSIZLP	*641L	911L															
177140	MTSIZLP0	*771L	731L															
177615	MTSIZMPF	*2981L	431L															
167531	MTTFLG	*681B																
167532	MTTLEN	*691B																
167533	MTTPNT	*701B																
000106	NECP	*43	1061J	1581J	3151J	4171J	4521J	4601J	7741J									
174247	NEWADR	*4131J	1231J															
174253	NOSADR	*4241J	5981J															
153611	OLDREGS	*69	1571J	9381J														
153607	OLDTOS	*68	2811F	3431F	121J	5591J	7511J	7611J	7871J	9221J								
170040	ONEMSA	*501F	1831F	1841F														
173020	ONLINXIT	*1581I	1551I															
000052	OPCODEBP	*44	5771J	5921J														
000000	ORIGINI	*24	71	74	81	641J	2051J	2091J	2591J	2621J	4191J	4851J	10331J					

[illegible]

[illegible]

[illegible]

175363	SWITZAF	*974:J	925:J					
000040	SWRPT	*64:IC	68:IC					
000045	SWSCF	*68:IC						
000020	SWSTDY	*63:IC						
000004	SWUSER	*61:IC	68:IC	735:J	772:J			
000010	SXABCL	*40:IC						
000002	SXADLO	*38:IC						
000004	SXADSC	*39:IC						
000001	SXAPND	*37:IC						
000200	SXAPWR	*44:IC						
000327	SXCLK	*37:ID						
147543	SXCRC	*39:ID						
000040	SXDCRQ	*31:IC						
000020	SXDPR	*30:IC						
000374	SXDTA	*36:ID						
173572	SXDUAL	*38:ID						
000040	SXMCD	*53:IC						
000010	SXMCTS	*51:IC						
000020	SXMDSR	*52:IC						
000100	SXMRNG	*54:IC						
000002	SXSOTR	*18:IC						
000010	SXSNSY	*21:IC						
000020	SXSRDY	*22:IC						
000001	SXSRQS	*17:IC						
170614	SYSCAM	*234:IF	325:IF					
171050	SYSCAT	*324:IF	185:IF	186:IF				
157000	SYSCOM	*34:IB						
167400	SYSESR	*31:IB	32:IB					
167400	SYSIVR	*32:IB	44:IB	128:IB				
170674	SYSPAT	*251:IF	38:IF					
170432	SYSPRM	*221:IF	268:IF					
160000	SYSRAM	*33:IB						
170000	SYSROM	*30:IB	134:IB					
175045	TDSTD	*804:J	802:J					
175060	TDWAIT	*811:J	812:J					
170044	TIMOUT	*58:IF						
170414	TIMOUTM	*220:IF	59:IF					
173275	TOFFTL	*371:II	85:II	124:II	154:II			
173264	TOFFTLDE	*352:II	623:IF	625:IF				
171616	TRIANGLE	*523:IF	398:IF					
175037	TSTDSP	*797:J	254:J					
176760	TSTMEN	*14:IL	224:J					
171040	UAINST	*320:IF	189:IF	190:IF				
170622	UAINTM	*235:IF	321:IF					
171627	UMST	*536:IF	549:IF	1000:J	110:IL	113:IL	114:IL	235:IL
000015	UMSTL	*549:IF	1001:J	60:IL				
171077	UTRANT	*352:IF	360:IF	362:IF	379:IF	383:IF		
000020	UTRANTL	*360:IF	378:IF	384:IF	391:IF			
170270	VECTI	*167:IF	209:IF	1010:J				
000124	VECTIL	*209:IF	259:IF	1012:J				
171664	WRAPST	*575:IF	38:IL					
170545	WVIDLM	*229:IF	309:IF					

171010 WVIDLT

*3081F

1771F

1781F